Attention Everyone!

Attention Everyone!

Neural Machine Translation by Jointly Learning to Align and Translate

Neural Traduction Automatique par Conjointement Apprentissage Pour Aligner et Traduire

Dzmitry Bahdanau
KyungHyun Cho
Yoshua Bengio
@ICLR 2015

Presented By Roee Aharoni

Haseminar Haorginal Mispar Ahat Mispar Arba

The Machine Translation Objective

e = (Economic, growth, has, slowed, down, in, recent, years, .)

f = (La, croissance, économique, s'est, ralentie, ces, dernières, années, .)

- We want to find the best translation **f** given a source sentence **e**

- $f = \underset{f'}{\text{argmax}}\ p(f'|e)$

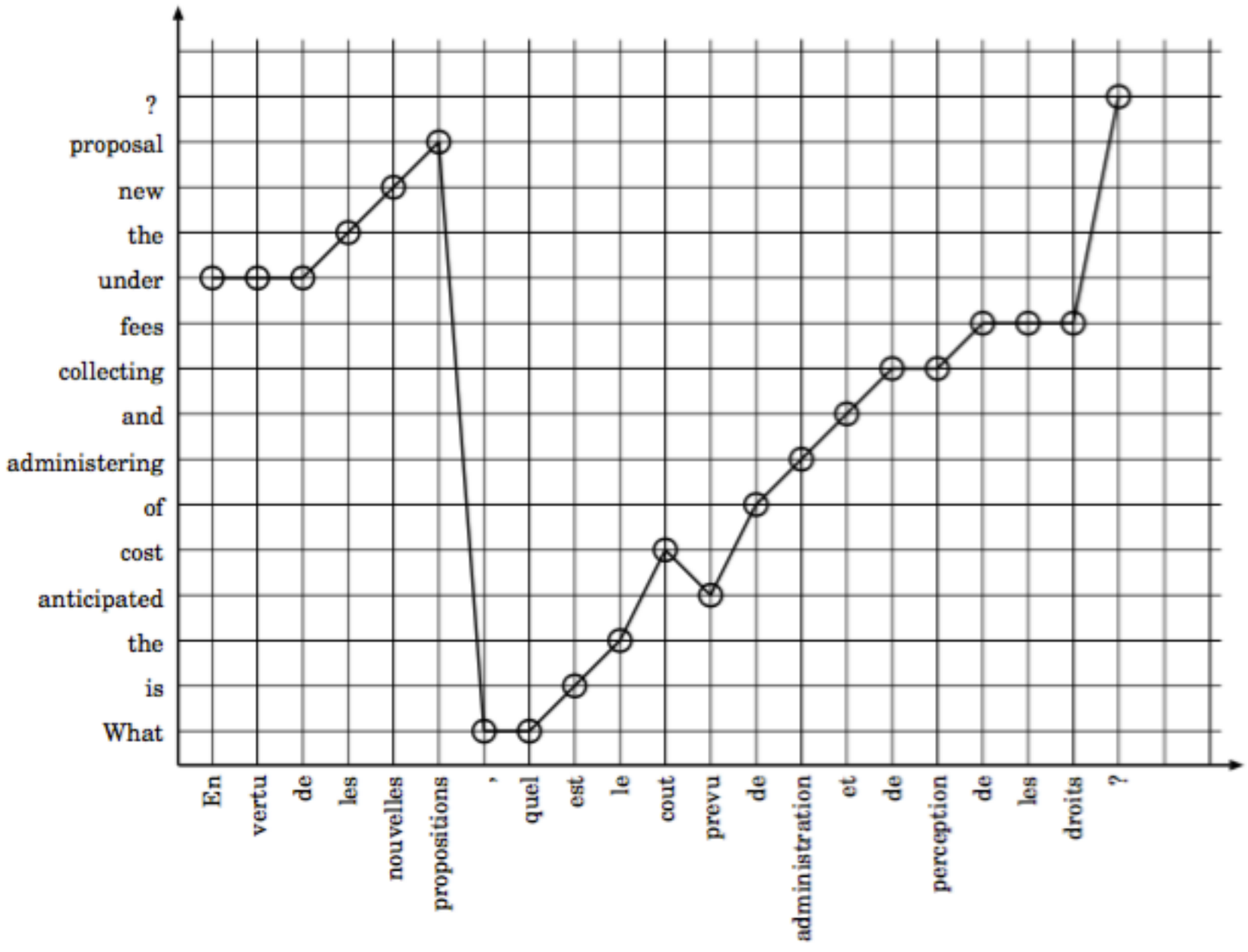
- How do we estimate p(f|e) from data?

“Conventional” Statistical Machine Translation

Start with (lots) of parallel text:

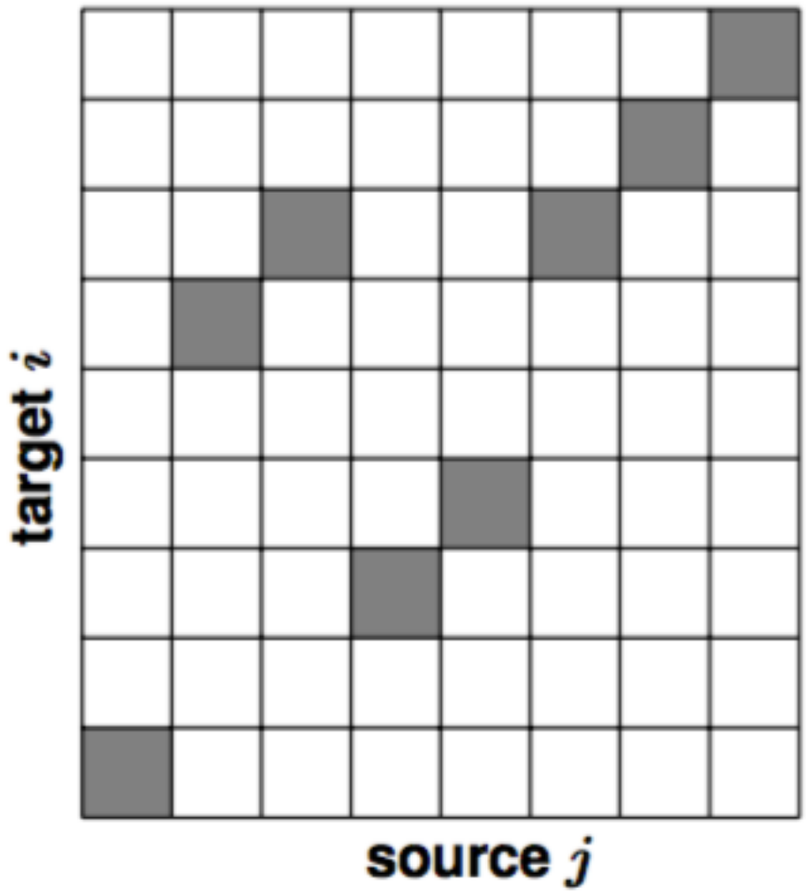
“Conventional” Statistical Machine Translation

Learn the alignments (using the EM algorithm):



“Conventional” Statistical Machine Translation

Learn the alignments (using the EM algorithm):



mapping: $j \rightarrow i = a_j$

“Conventional” Statistical Machine Translation

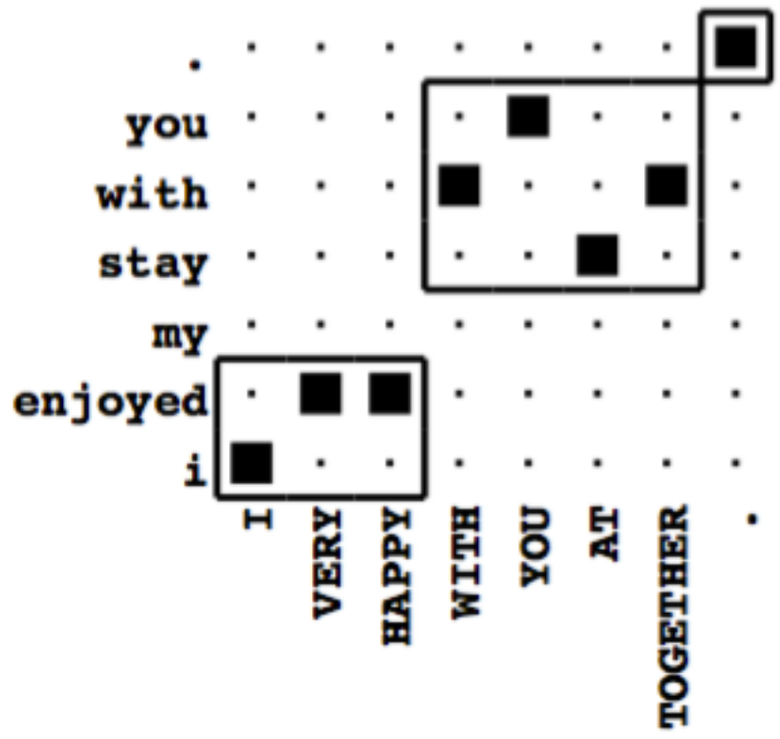
Extract phrase pairs:

source sentence 我 很 高兴 和 你 在 一起 .

gloss notation I VERY HAPPY WITH YOU AT TOGETHER .

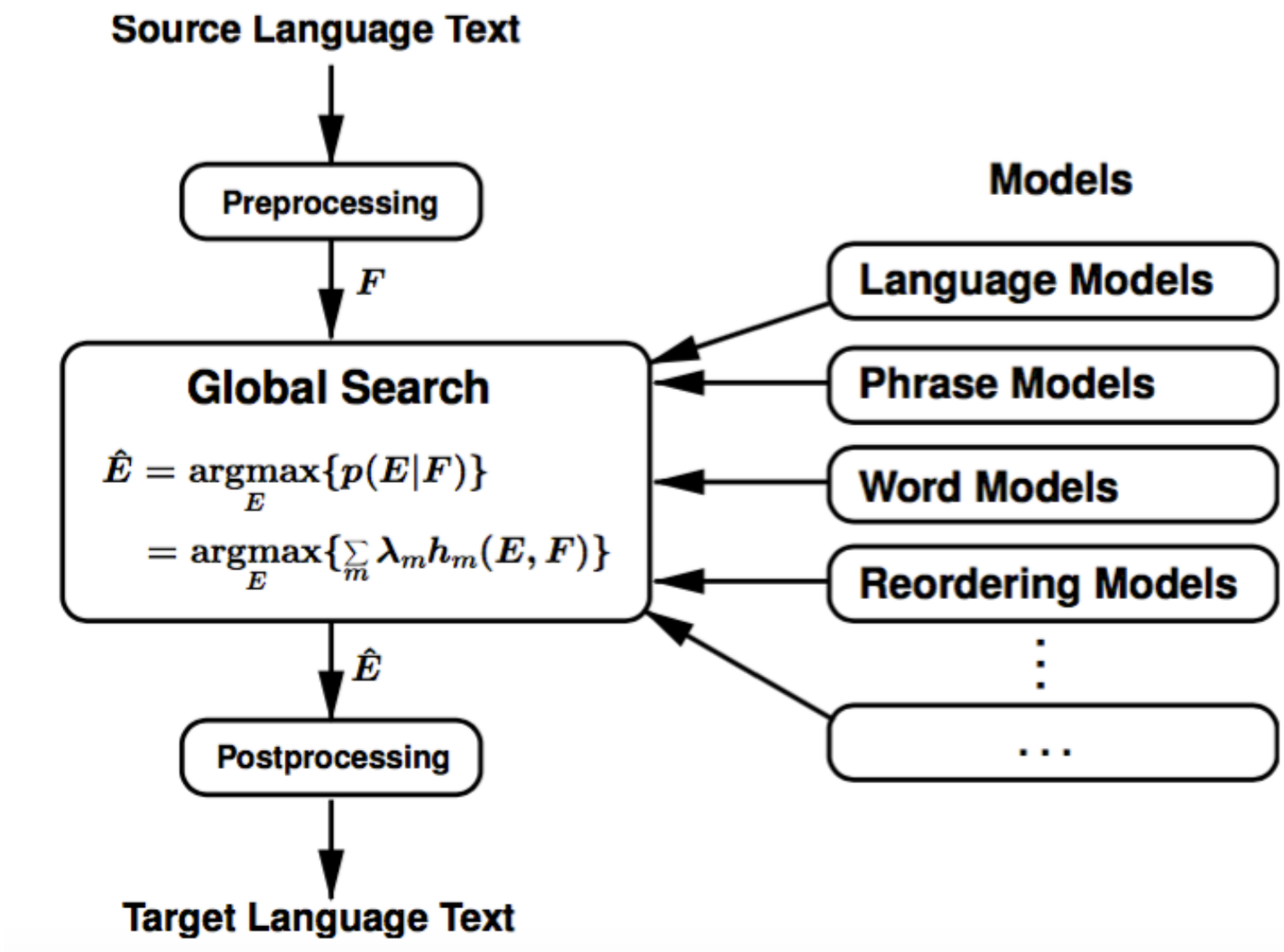
target sentence I enjoyed my stay with you .

Viterbi alignment for $F \rightarrow E$:



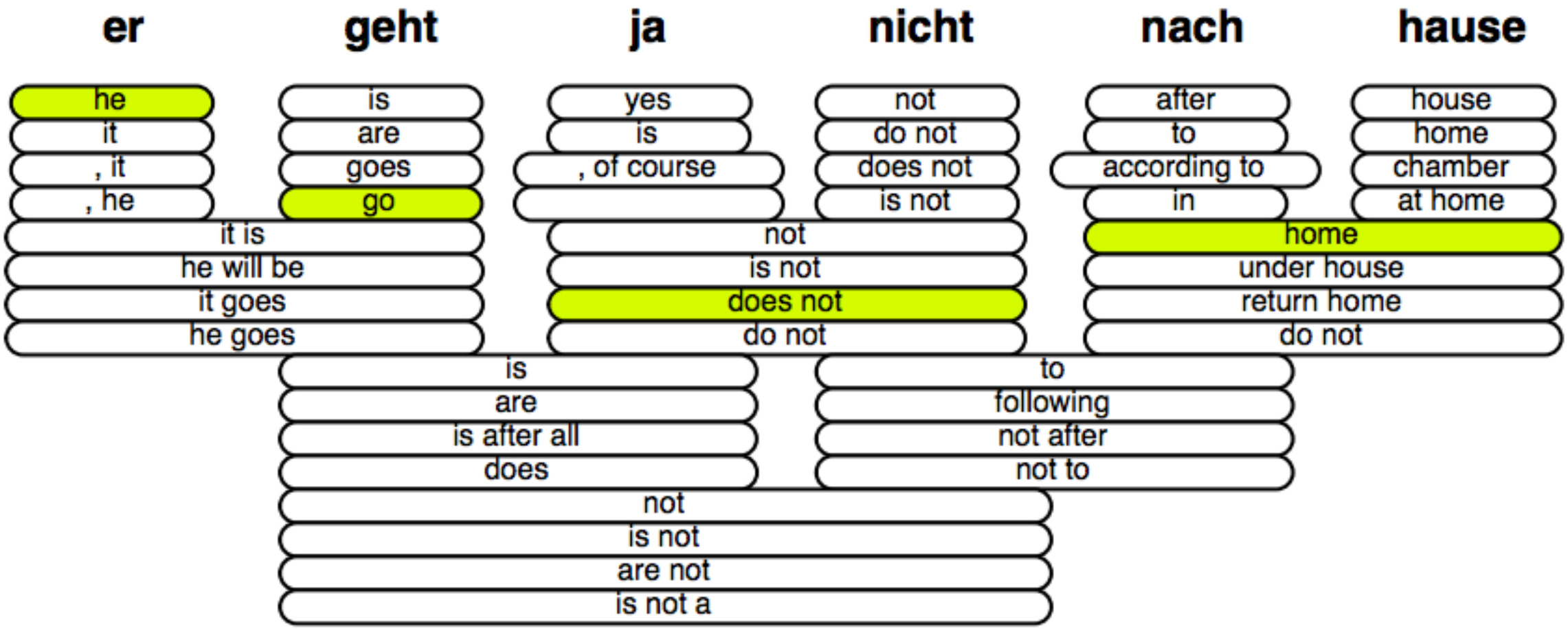
“Conventional” Statistical Machine Translation

Use a log linear model combination to **score** possible hypotheses:



“Conventional” Statistical Machine Translation

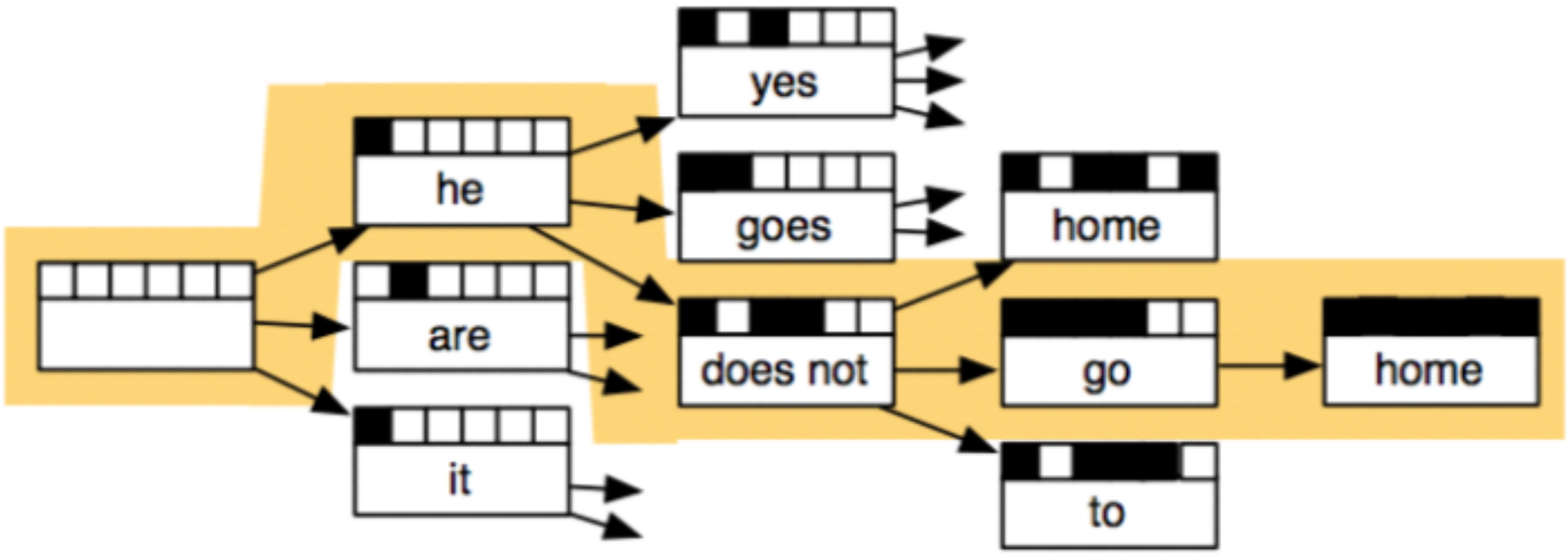
Use a log linear model combination to score **possible hypotheses**:



- Many translation options to choose from
  - in Europarl phrase table: 2727 matching phrase pairs for this sentence
  - by pruning to the top 20 per phrase, 202 translation options remain

“Conventional” Statistical Machine Translation

Use beam search to output the best hypothesis:



backtrack from highest scoring complete hypothesis

So far we had…

So far we had…

- Learn Alignments from a large parallel corpus

So far we had…

- Learn Alignments from a large parallel corpus
- Extract phrases from the alignments

So far we had…

- Learn Alignments from a large parallel corpus
- Extract phrases from the alignments
- Extract some more features

So far we had…

- Learn Alignments from a large parallel corpus
- Extract phrases from the alignments
- Extract some more features
- (And then some more…)

So far we had…

- Learn Alignments from a large parallel corpus
- Extract phrases from the alignments
- Extract some more features
- (And then some more…)
- Train one combined model using all the above + large language model: estimate **p(fle) as p(elf)p(f)**

So far we had…

- Learn Alignments from a large parallel corpus
- Extract phrases from the alignments
- Extract some more features
- (And then some more…)
- Train one combined model using all the above + large language model: estimate **p(fle) as p(elf)p(f)**

translation

model

So far we had…

- Learn Alignments from a large parallel corpus
- Extract phrases from the alignments
- Extract some more features
- (And then some more…)
- Train one combined model using all the above + large language model: estimate **p(fle) as p(elf)p(f)**

translation model

language model

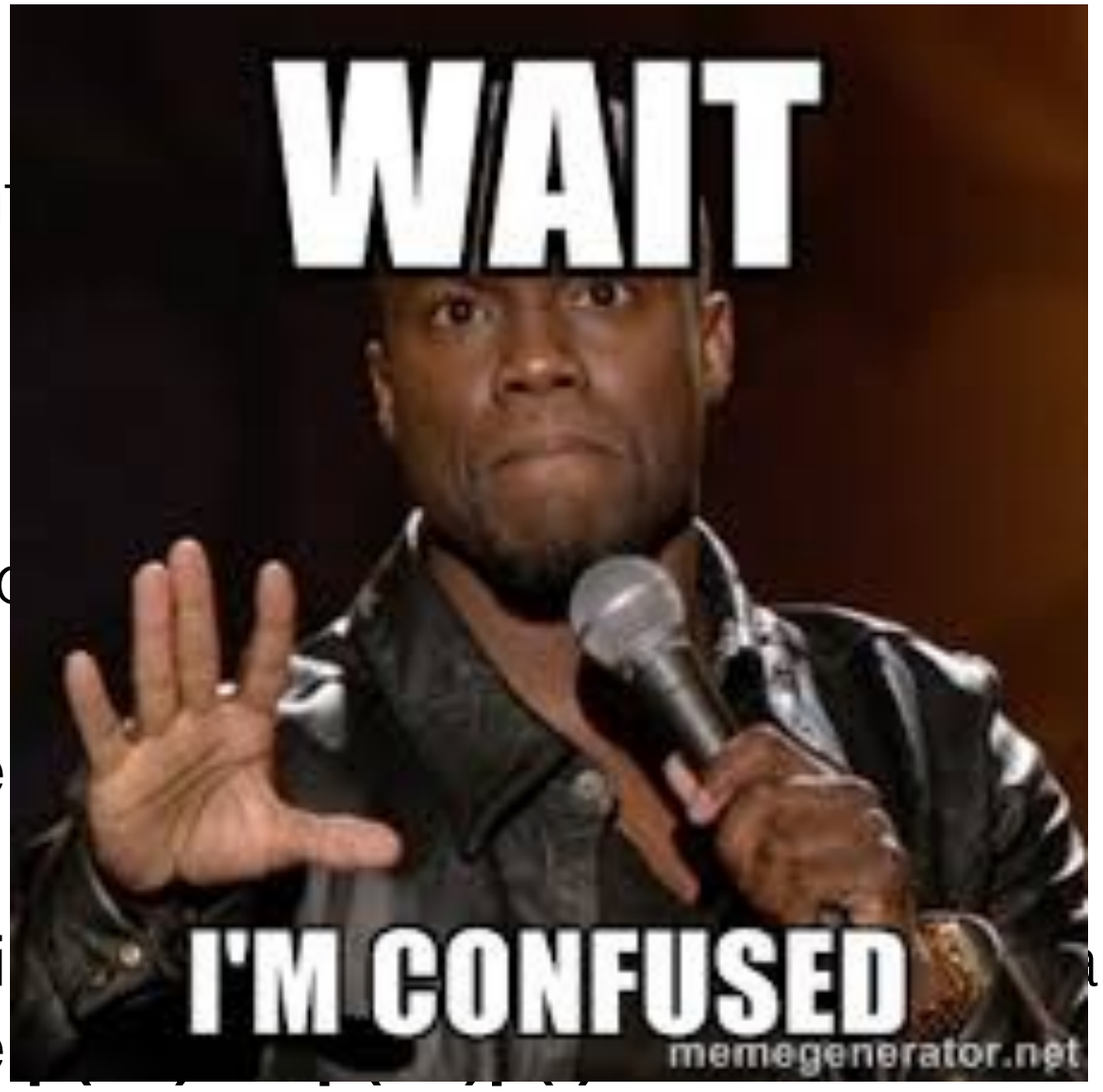
So far we had…

- Learn Alignments from a large parallel corpus
- Extract phrases from the alignments
- Extract some more features
- (And then some more…)
- Train one combined model using all the above + large language model: estimate **p(fle) as p(elf)p(f)**

  translation model × language model
- Run search on top of that

So far we had…

- Learn Alignment
- Extract phrases
- Extract some mo
- (And then some
- Train one combi rge language model: estimate
- Run search on top of that



translation model

language model

So let’s take another approach…

So let's take another approach…

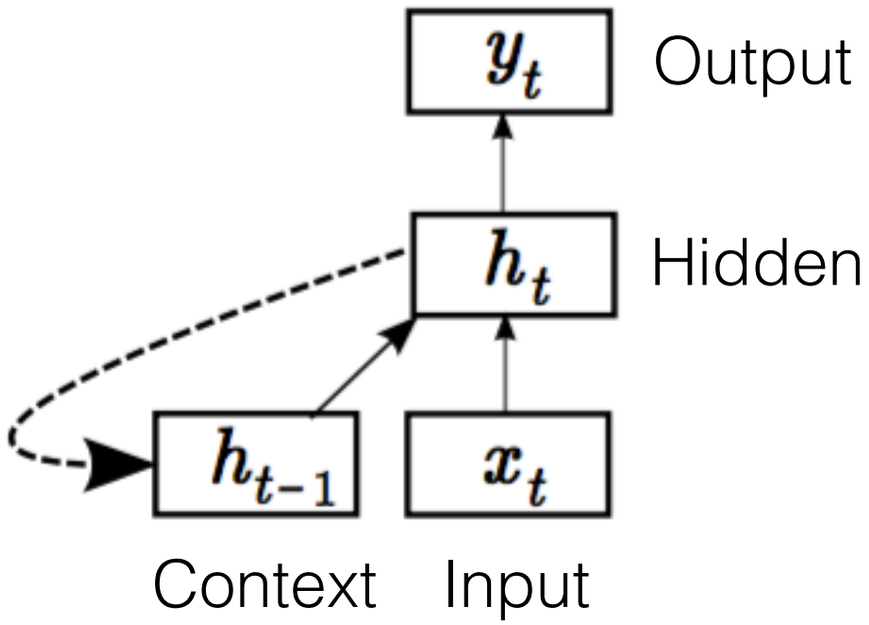
- Maybe we can just estimate **p(fle)** directly?

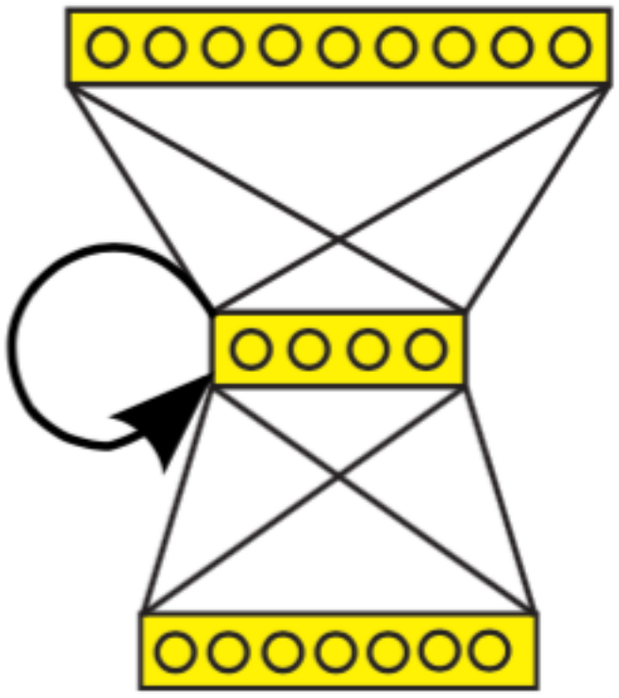
So let's take another approach…

- Maybe we can just estimate **p(fle)** directly?
- First, let's get familiar with -

Recurrent Neural Networks (RNN's)

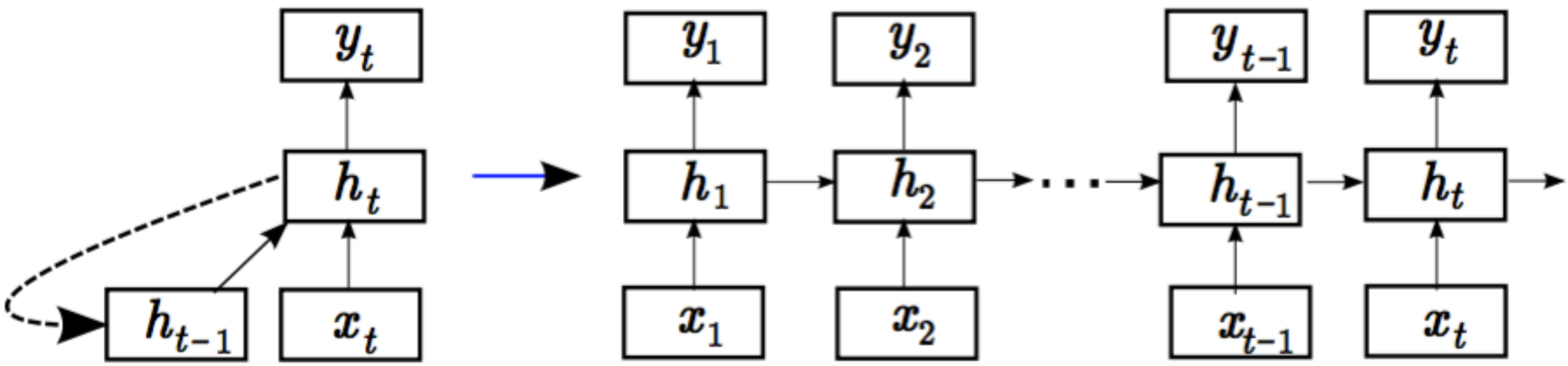
- Enable variable length inputs (sequences)
- Sensitive to the order of the elements in the input
- Introduce a "memory/context" component to utilize history





Recurrent Neural Networks (RNN's)

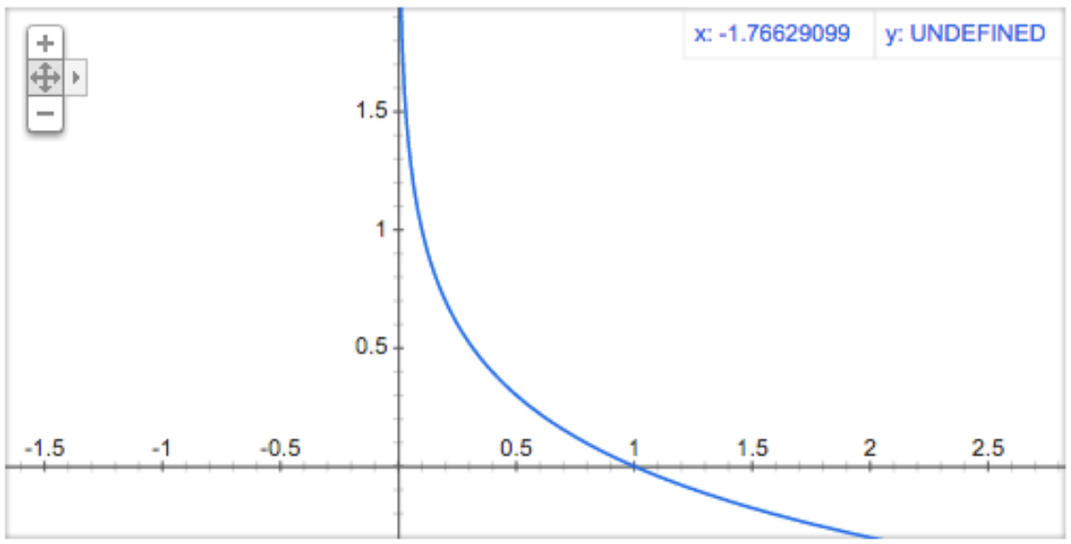
- "Horizontally deep" architecture
- Recurrence equations:
  - Transition function: $h_t = H(h_{t-1}, x_t) = tanh(Wx_{t-1} + Uh_{t-1} + b)$
  - Output function: $y_t = Y(h_t)$, commonly defined as softmax



The Softmax Function & Neg. Log. Loss

- Enables to output a probability distribution over k possible classes
- y_i (the value of the network output vector on index i) is expected to hold log-likelihood of a specific class (in our case, word):
- Our loss function will be the sum of negative log softmax values over the correct sequence

Graph for -log(x)



$$p(x = i) = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}}$$

Training (RNN's) with Backpropagation Through Time

- As usual, define a loss function (per sample, through time $t = 1, 2, ..., T$):

$$Loss = J(\Theta, x) = -\sum_{t=1}^{T} J_t(\Theta, x_t)$$

- Derive the loss function w.r.t. parameters Θ, starting at $t = T$:

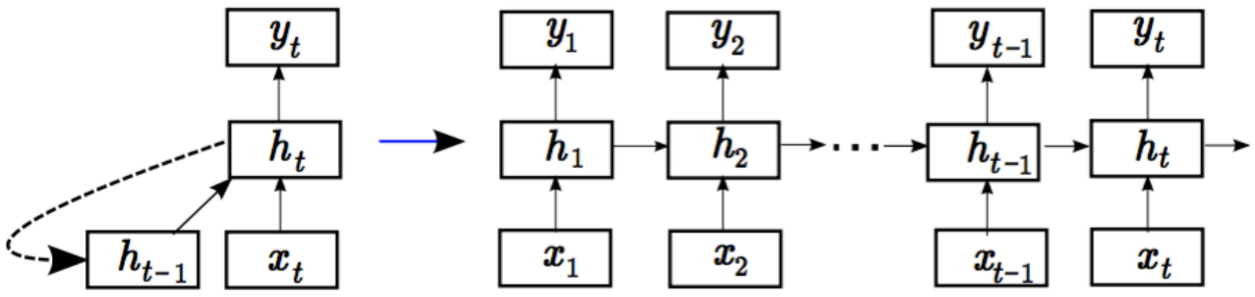
$$\nabla\Theta = \frac{\partial J_t}{\partial \Theta}$$

- Backpropagate through time - update and repeat for $t - 1$, until $t = 1$:

$$\nabla\Theta = \nabla\Theta + \frac{\partial J_t}{\partial \Theta}$$

- Eventually, update the weights:

$$\Theta = \gamma\nabla\Theta$$



LSTM walkthrough in 4 steps

- Processes a variable length input sequence: $x = (x_1, x_2, \cdots, x_n)$
- In each time step, holds a memory cell c_t and a hidden state h_t used for predicting an output
- Has gates controlling the extent to which new content should be memorized (**input gate**), old content should be erased (**forget gate**), and current content should be exposed (**output gate**). More formally:

LSTM walkthrough in 4 steps

- Processes a variable length input sequence: $x = (x_1, x_2, \cdots, x_n)$
- In each time step, holds a memory cell c_t and a hidden state h_t used for predicting an output
- Has gates controlling the extent to which new content should be memorized (**input gate**), old content should be erased (**forget gate**), and current content should be exposed (**output gate**). More formally:

I

compute input, forget, output gates and memory cell update

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

LSTM walkthrough in 4 steps

- Processes a variable length input sequence: $x = (x_1, x_2, \cdots, x_n)$
- In each time step, holds a memory cell c_t and a hidden state h_t used for predicting an output
- Has gates controlling the extent to which new content should be memorized (**input gate**), old content should be erased (**forget gate**), and current content should be exposed (**output gate**). More formally:

I compute input, forget, output gates and memory cell update

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

II compute current memory cell using input and forget gates

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

LSTM walkthrough in 4 steps

- Processes a variable length input sequence: $x = (x_1, x_2, \cdots, x_n)$
- In each time step, holds a memory cell c_t and a hidden state h_t used for predicting an output
- Has gates controlling the extent to which new content should be memorized (**input gate**), old content should be erased (**forget gate**), and current content should be exposed (**output gate**). More formally:

I compute input, forget, output gates and memory cell <u>update</u>

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

II compute current memory cell using input and forget gates

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

III

$$h_t = o_t \odot \tanh(c_t)$$

compute current hidden state using output gate and memory cell

LSTM walkthrough in 4 steps

- Processes a variable length input sequence: $x = (x_1, x_2, \cdots, x_n)$
- In each time step, holds a memory cell c_t and a hidden state h_t used for predicting an output
- Has gates controlling the extent to which new content should be memorized (**input gate**), old content should be erased (**forget gate**), and current content should be exposed (**output gate**). More formally:

I compute input, forget, output gates and memory cell update

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

II compute current memory cell using input and forget gates

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

III compute current hidden state using output gate and memory cell

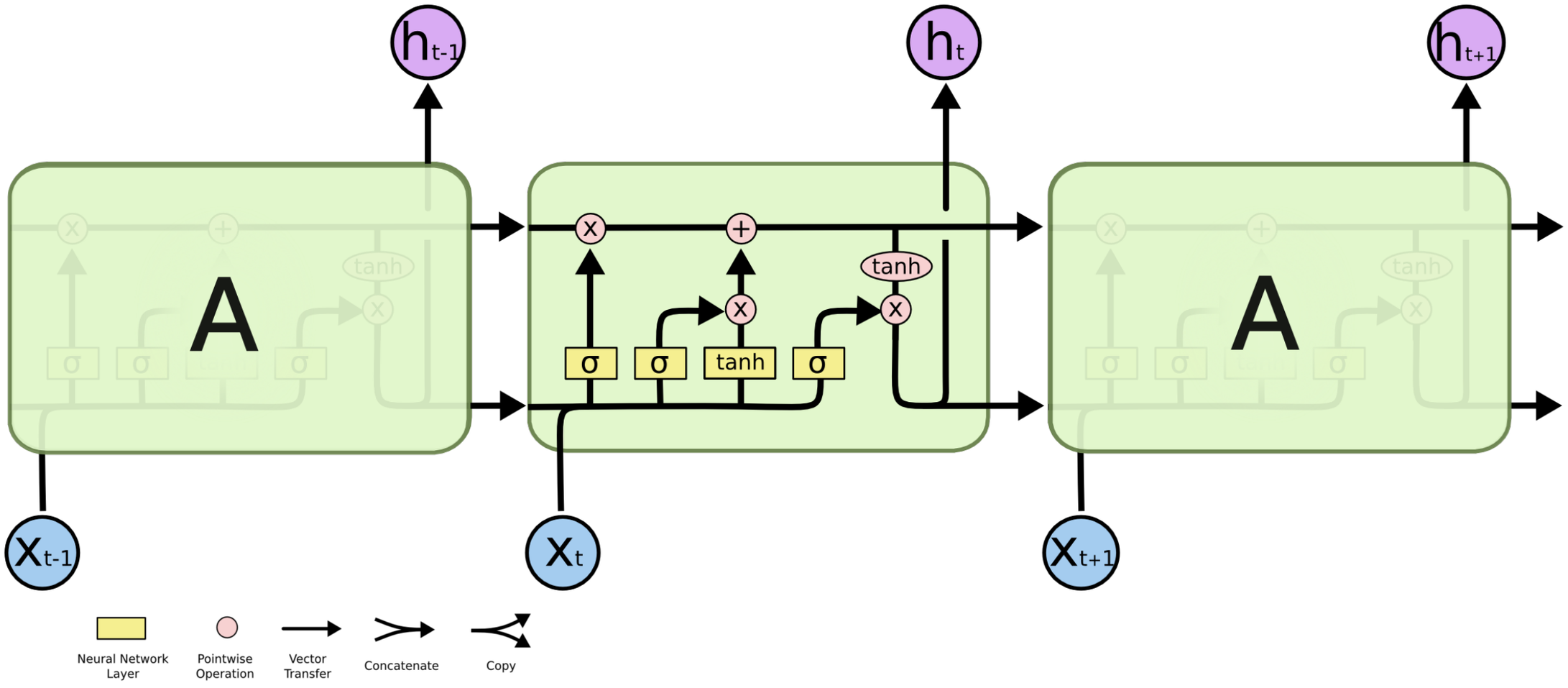
$$h_t = o_t \odot \tanh(c_t)$$

IV compute current output probabilities for prediction by using softmax over the hidden state

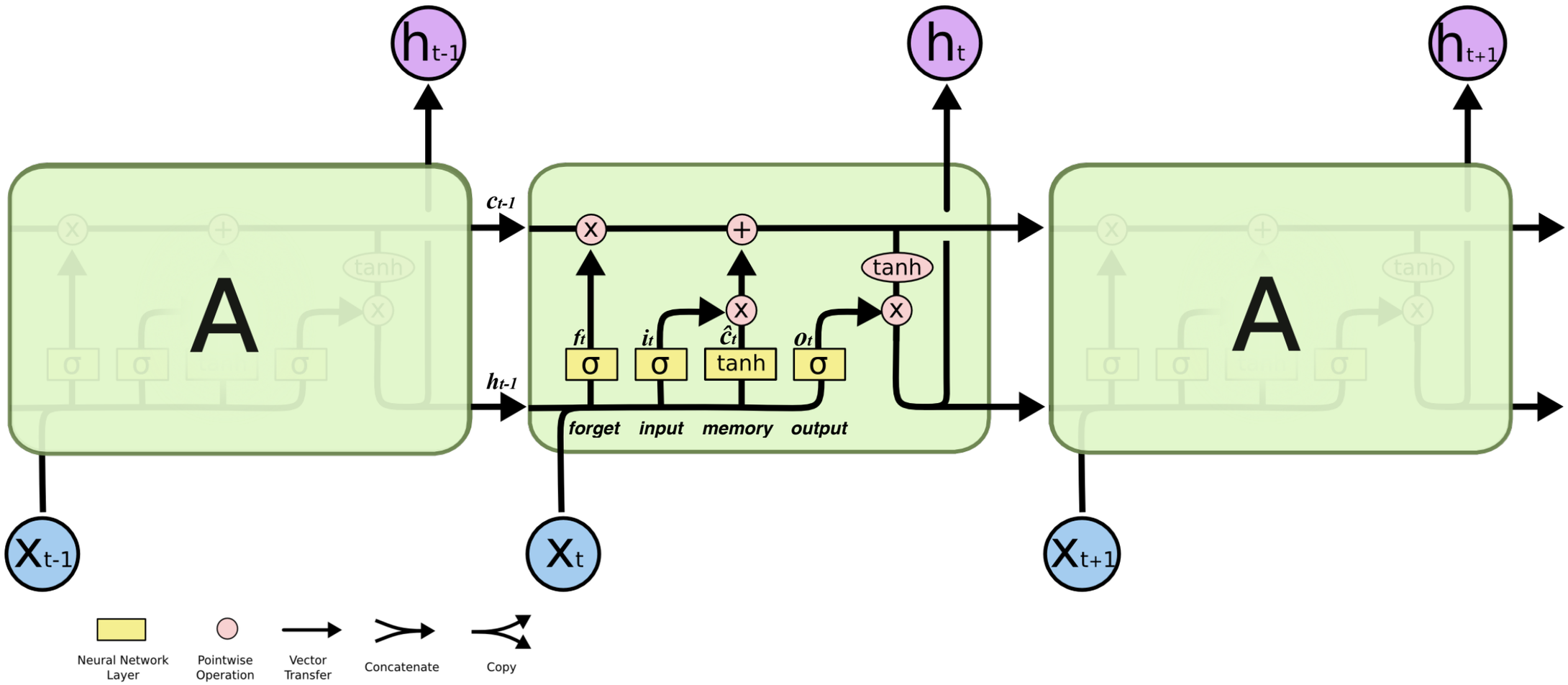
$$p(x_{t+1} = w | x_1, \cdots, x_t) = \exp(u(w, h_t))/Z$$

$$Z = \sum_{w' \in V} \exp(u(w', h_t))$$

LSTM walkthrough in 4 steps



LSTM walkthrough in 4 steps

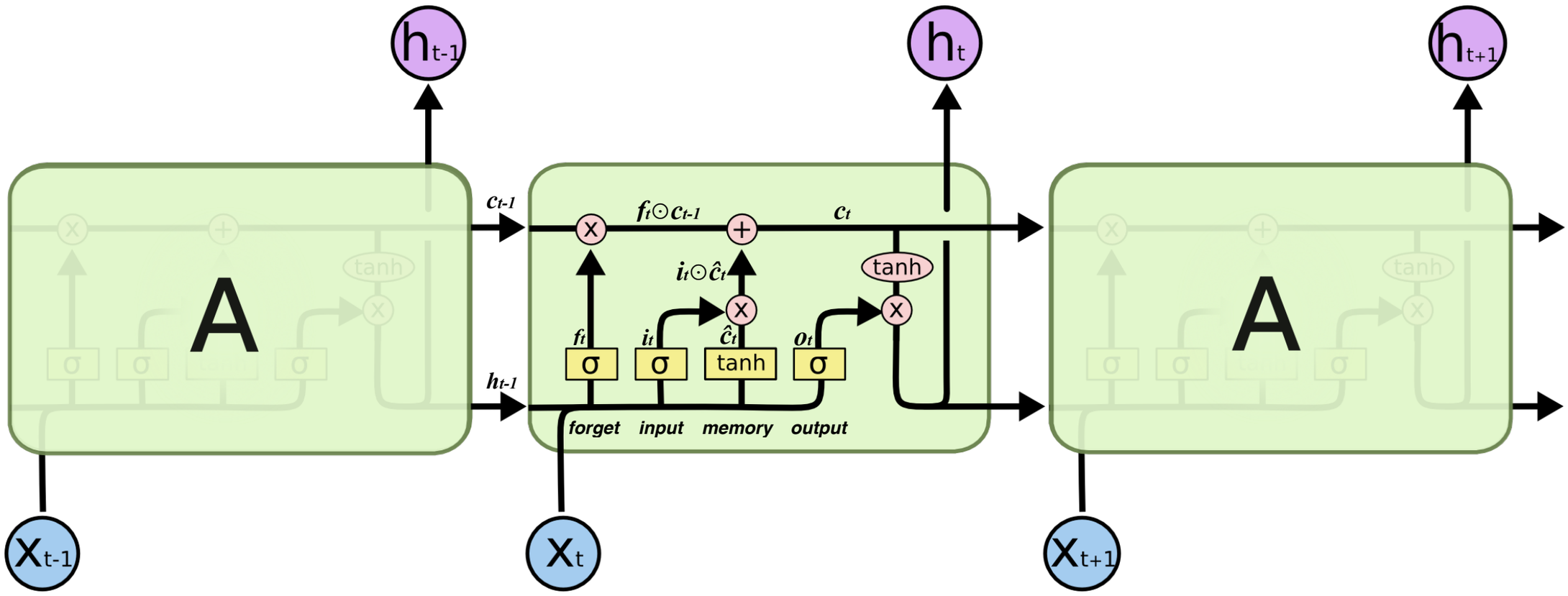


I

compute current input, forget, output, memory gate values

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

LSTM walkthrough in 4 steps



Neural Network Layer · Pointwise Operation · Vector Transfer · Concatenate · Copy

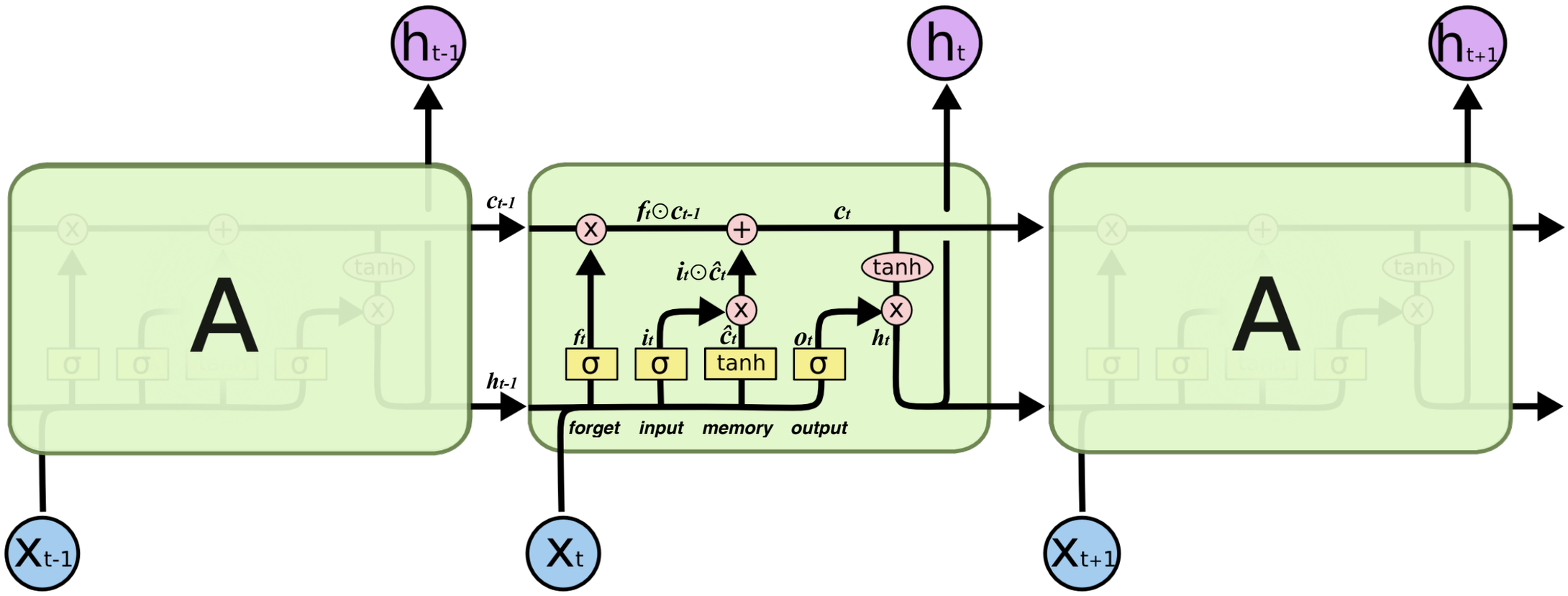
II compute current memory cell using input and forget gates

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

I compute current input, forget, output, memory gate values

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

LSTM walkthrough in 4 steps



Neural Network Layer · Pointwise Operation · Vector Transfer · Concatenate · Copy

I compute current input, forget, output, memory gate values

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

II compute current memory cell using input and forget gates

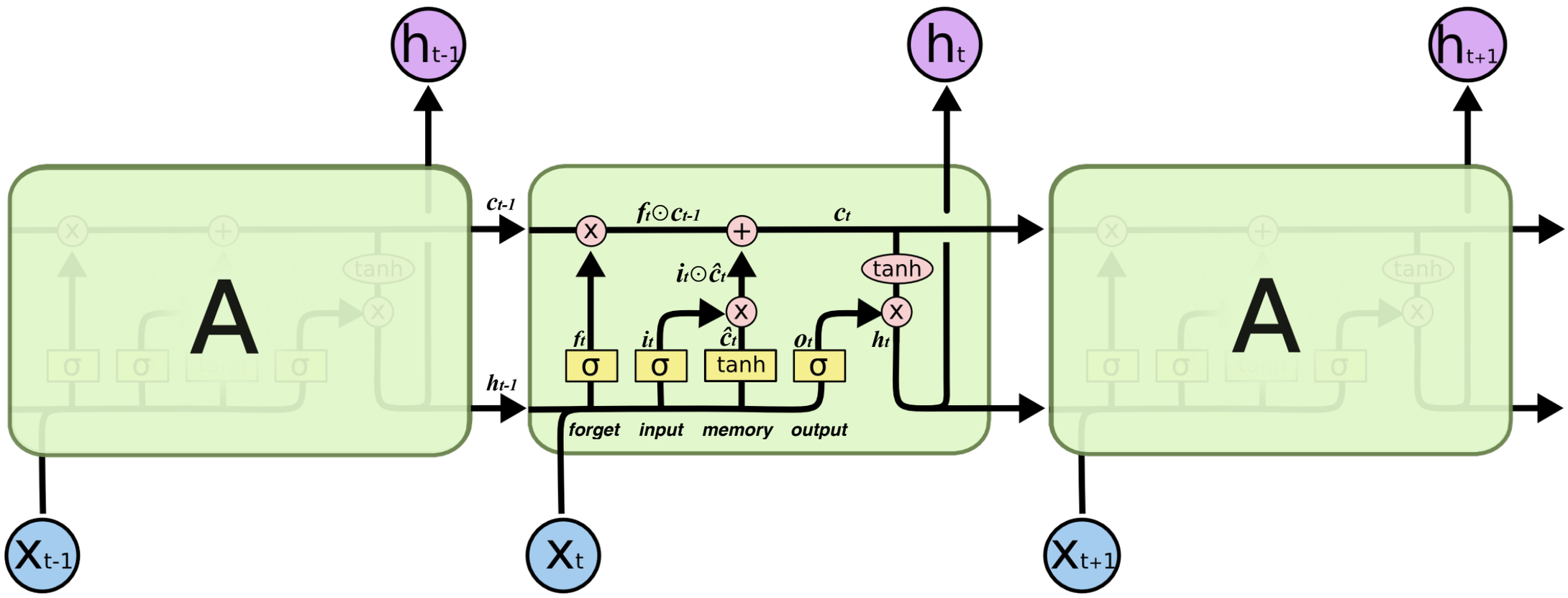
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

III compute current hidden state using output gate and memory cell

$$h_t = o_t \odot \tanh(c_t)$$

LSTM walkthrough in 4 steps

$$p(x_{t+1} = w|x_1, \cdots, x_t) = \exp(u(w, h_t))/Z$$



Neural Network Layer · Pointwise Operation · Vector Transfer · Concatenate · Copy

I compute current input, forget, output, memory gate values

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

II compute current memory cell using input and forget gates

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

III compute current hidden state using output gate and memory cell

$$h_t = o_t \odot \tanh(c_t)$$

IV

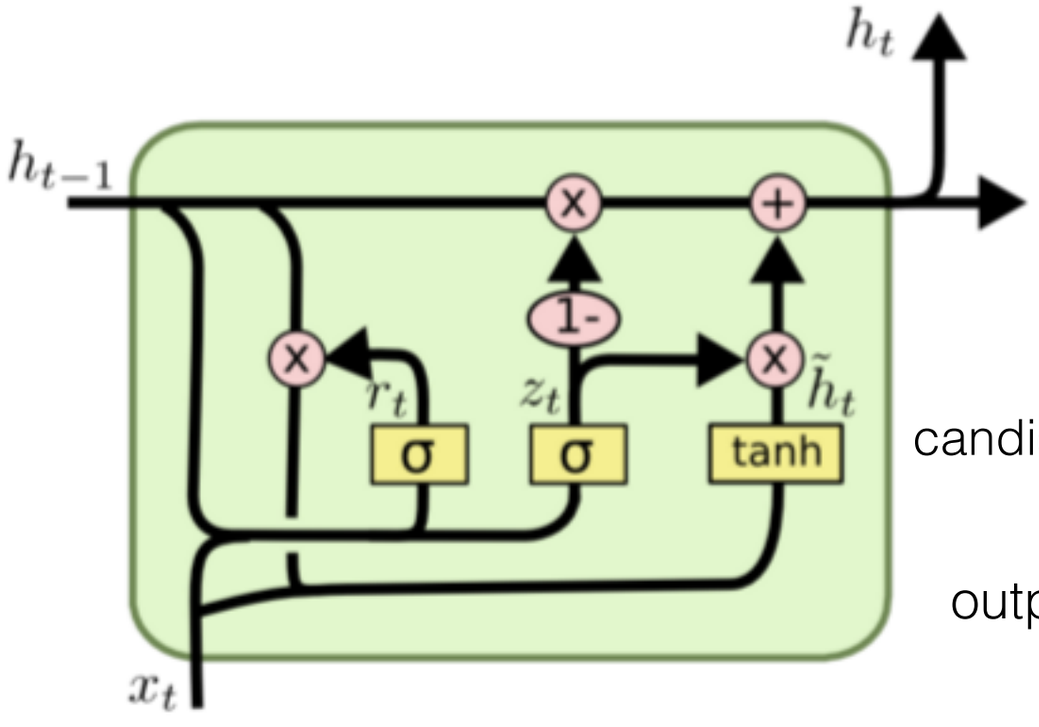
$$p(x_{t+1} = w|x_1, \cdots, x_t) = \exp(u(w, h_t))/Z$$

$$Z = \sum_{w' \in V} \exp(u(w', h_t))$$

compute current output probabilities for prediction by using softmax over the hidden state

The GRU

- “Gated Recurrent Unit”
- Simpler than LSTM
- ±Similar performance



the GRU architecture

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \quad \text{update gate}$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \quad \text{reset gate}$$

$$\text{candidate} \quad \tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$\text{output} \quad h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Sequence 2 Sequence Learning - Background

Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014

Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014

- Inspired by RNN language modeling

Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)

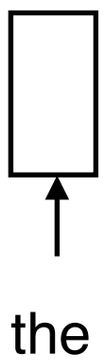
Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)

Encoder

Sequence 2 Sequence Learning - Background

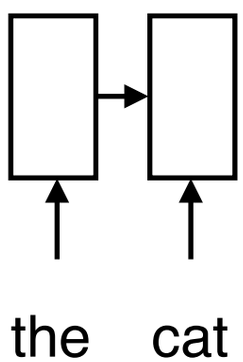
- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



Encoder

Sequence 2 Sequence Learning - Background

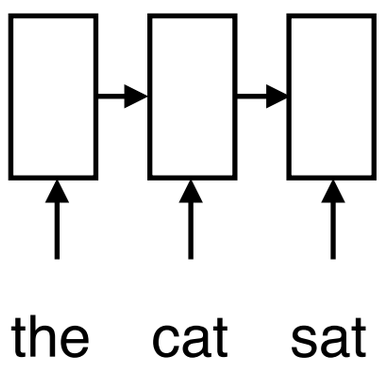
- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



Encoder

Sequence 2 Sequence Learning - Background

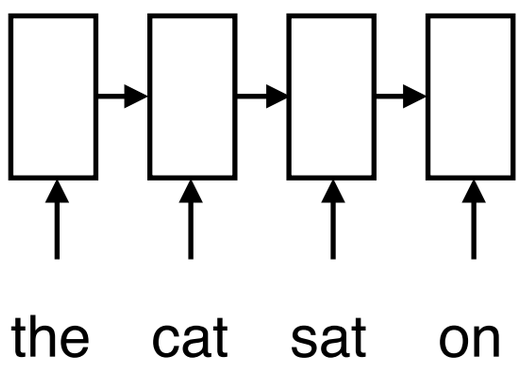
- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



Encoder

Sequence 2 Sequence Learning - Background

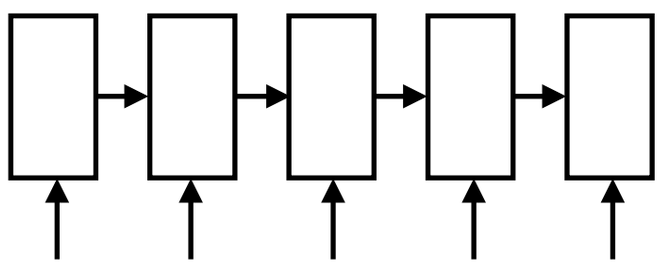
- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



Encoder

Sequence 2 Sequence Learning - Background

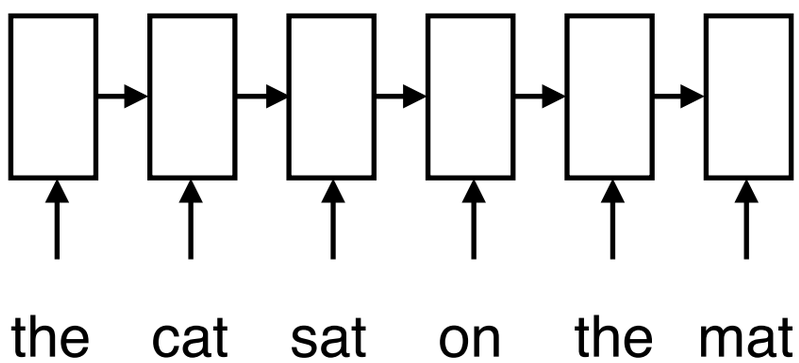
- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



Encoder

Sequence 2 Sequence Learning - Background

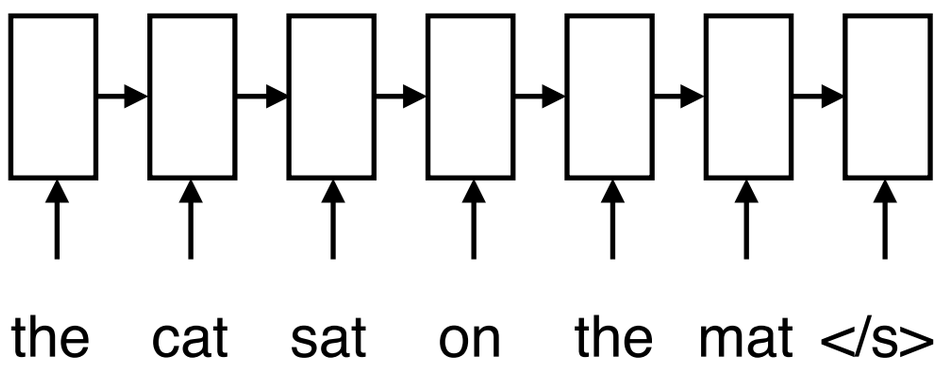
- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



Encoder

Sequence 2 Sequence Learning - Background

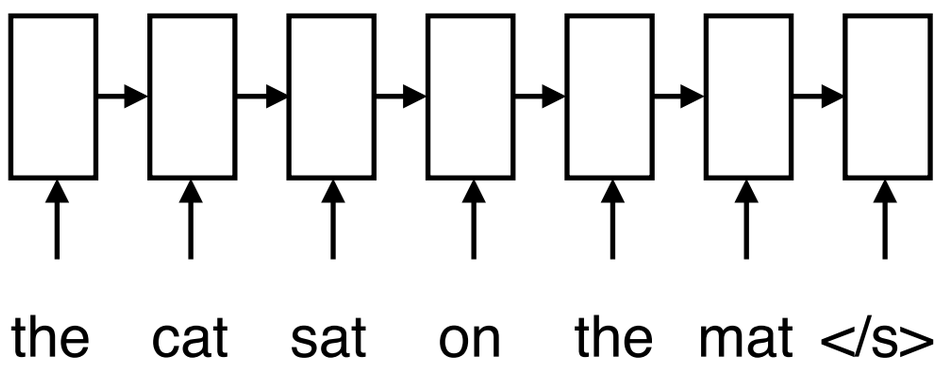
- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



Encoder

Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)

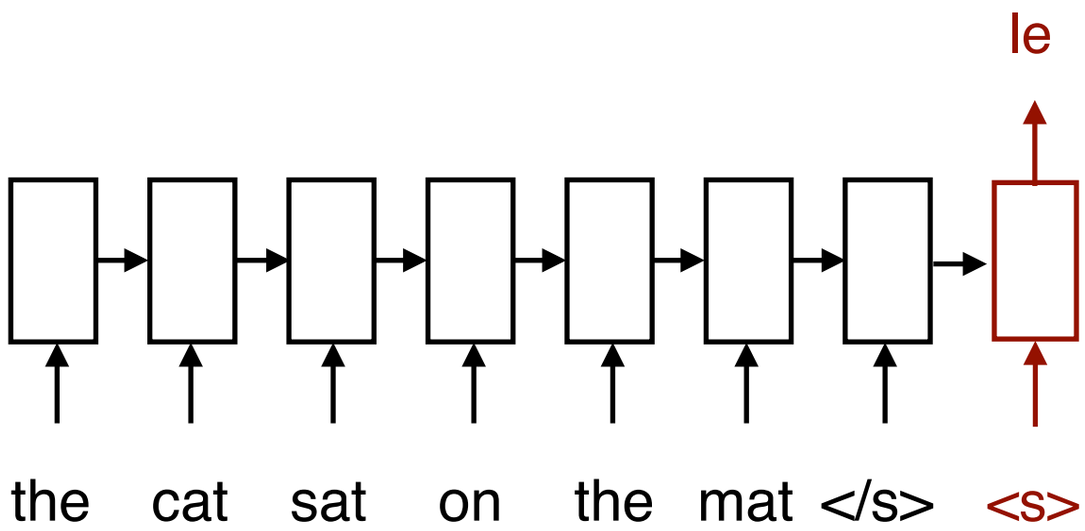


Encoder

Decoder

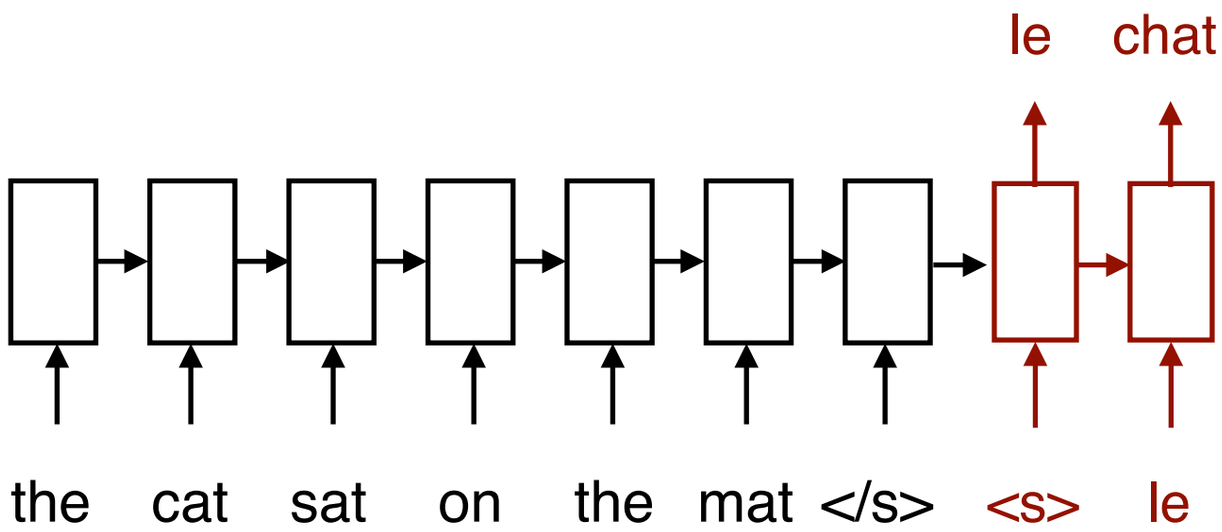
Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



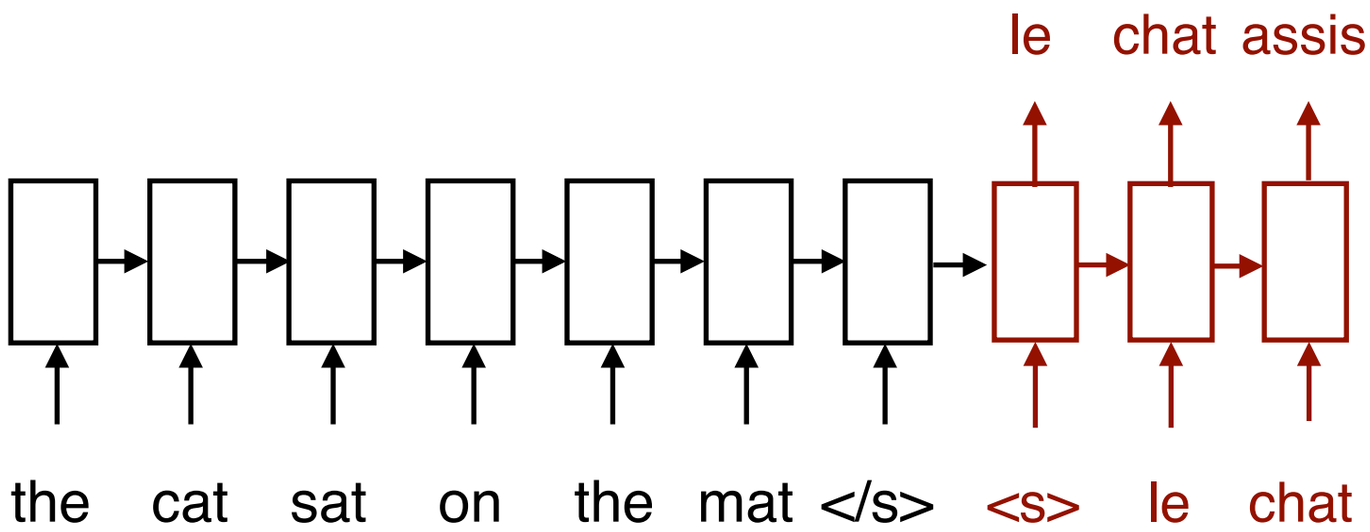
Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



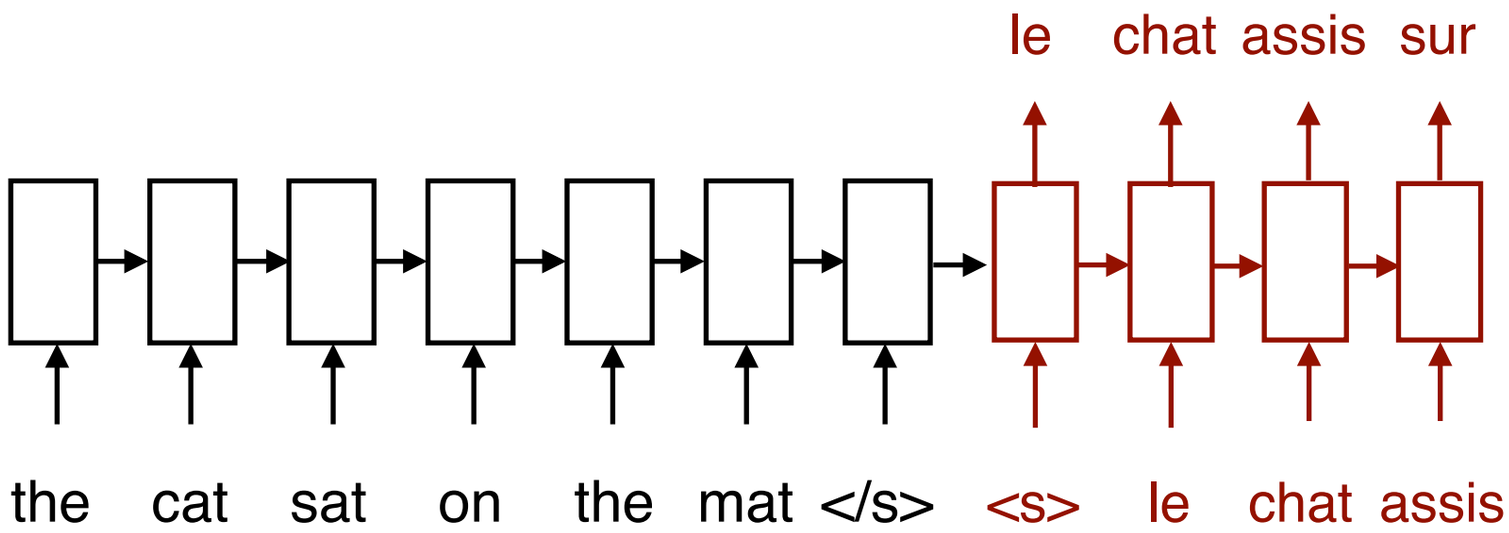
Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



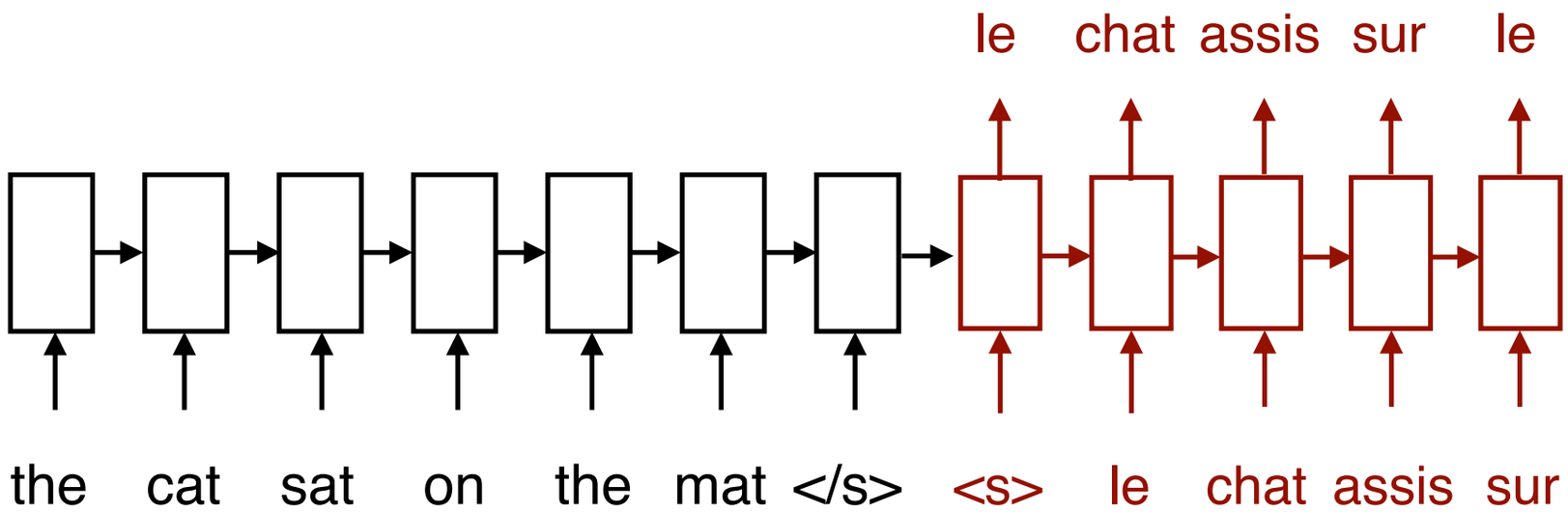
Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



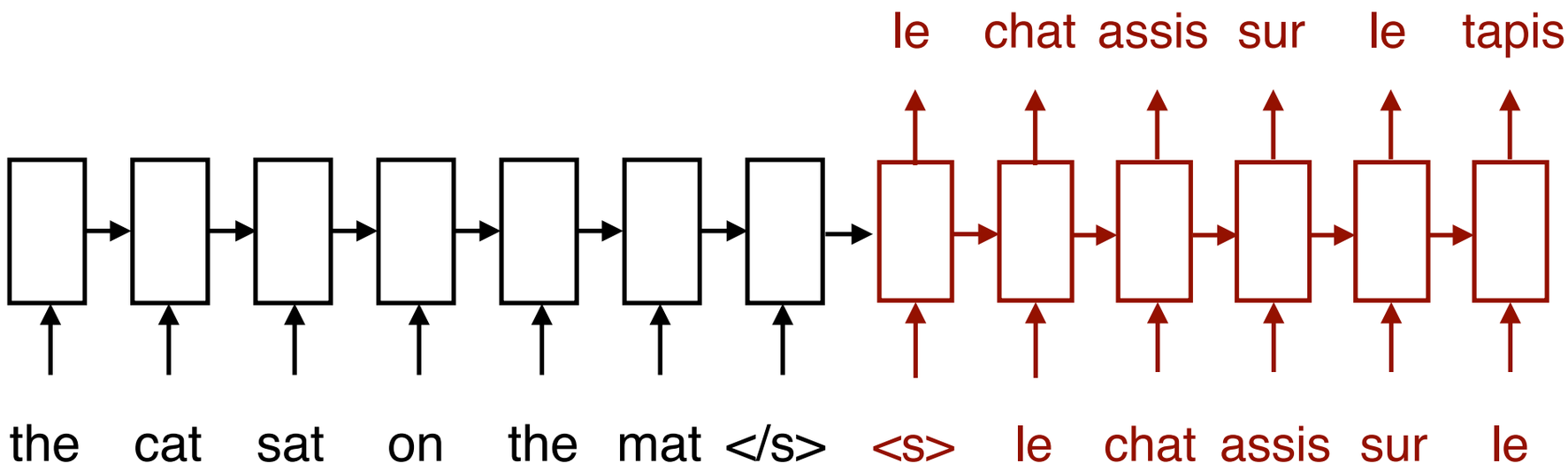
Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



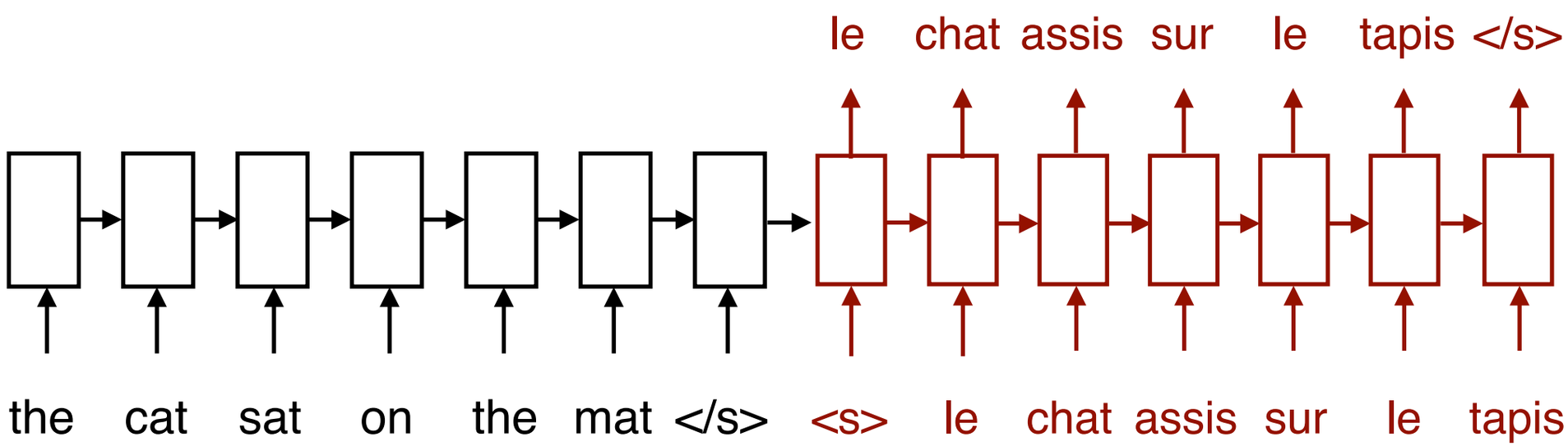
Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)

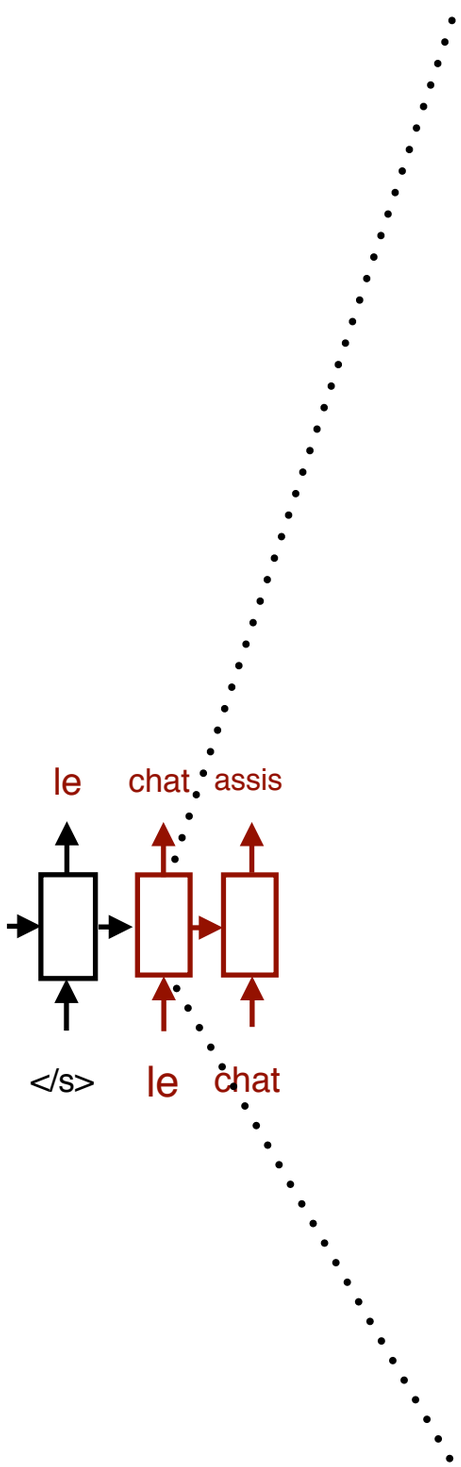


Sequence 2 Sequence Learning - Background

- First (modern) models for NMT presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modeling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture)



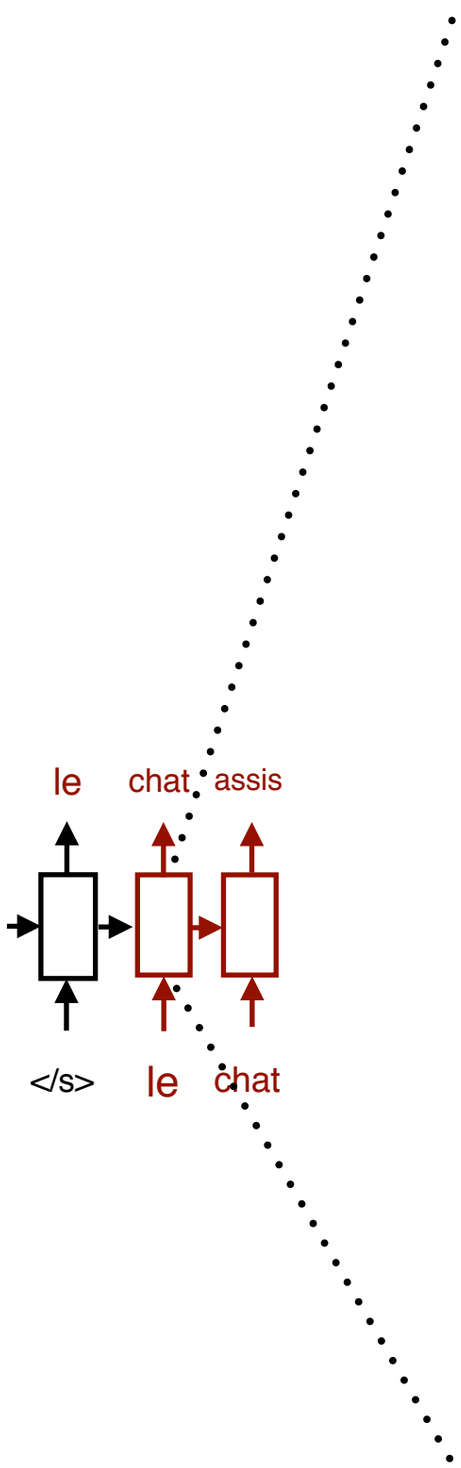
Seq2Seq decoder step - Zoom-In



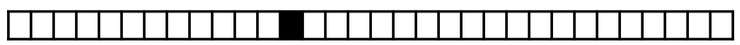
“chat”

“le”

Seq2Seq decoder step - Zoom-In



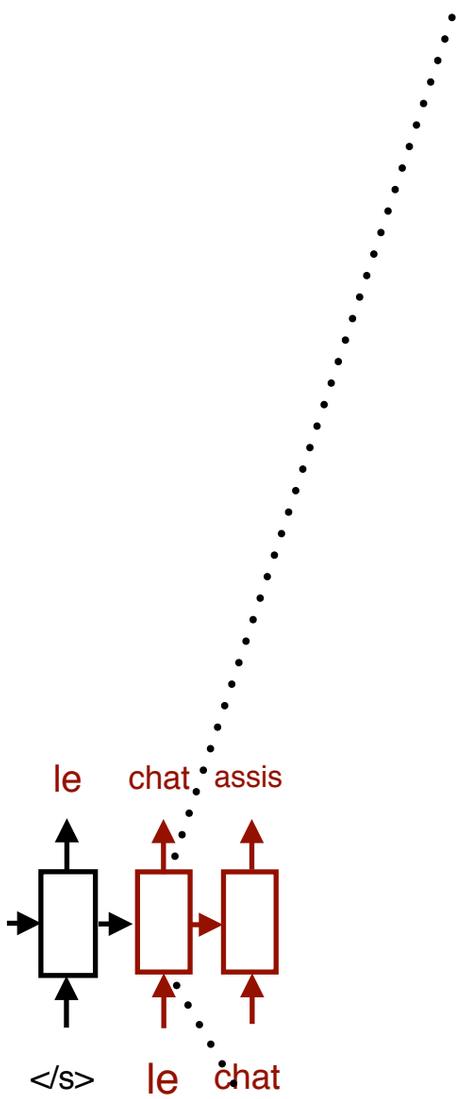
1-hot vec for symbol at time *t*



"le"

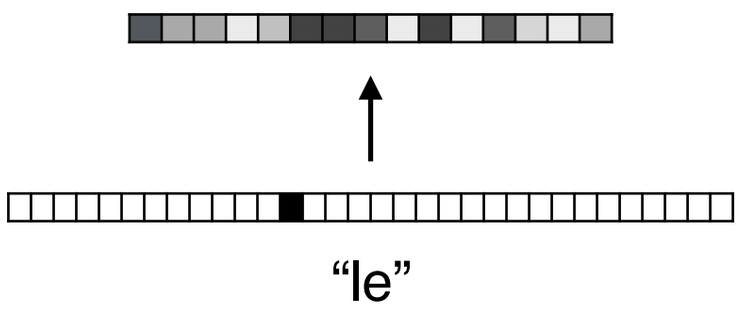
input vocabulary size

Seq2Seq decoder step - Zoom-In



input symbol embedding

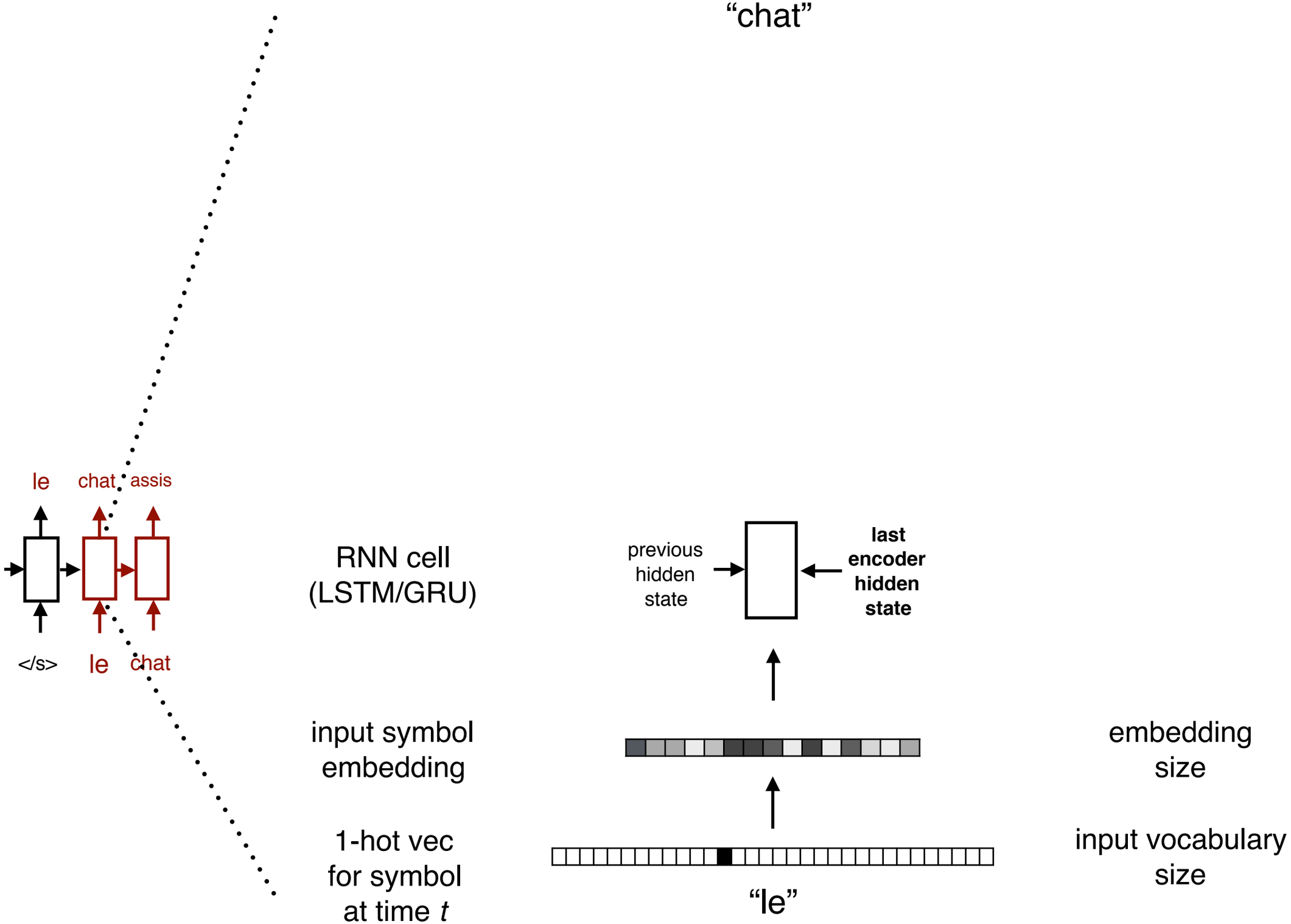
1-hot vec for symbol at time *t*



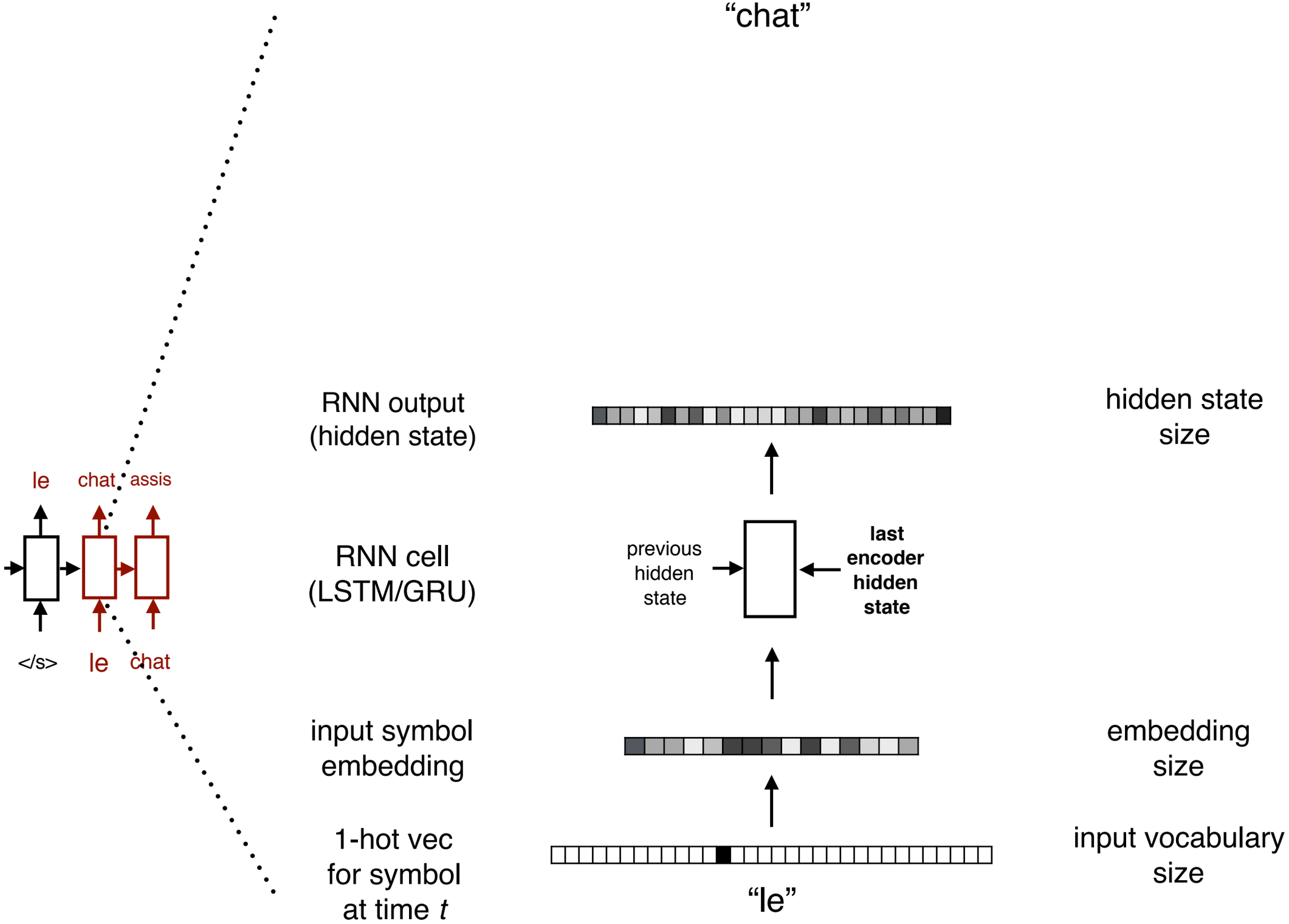
embedding size

input vocabulary size

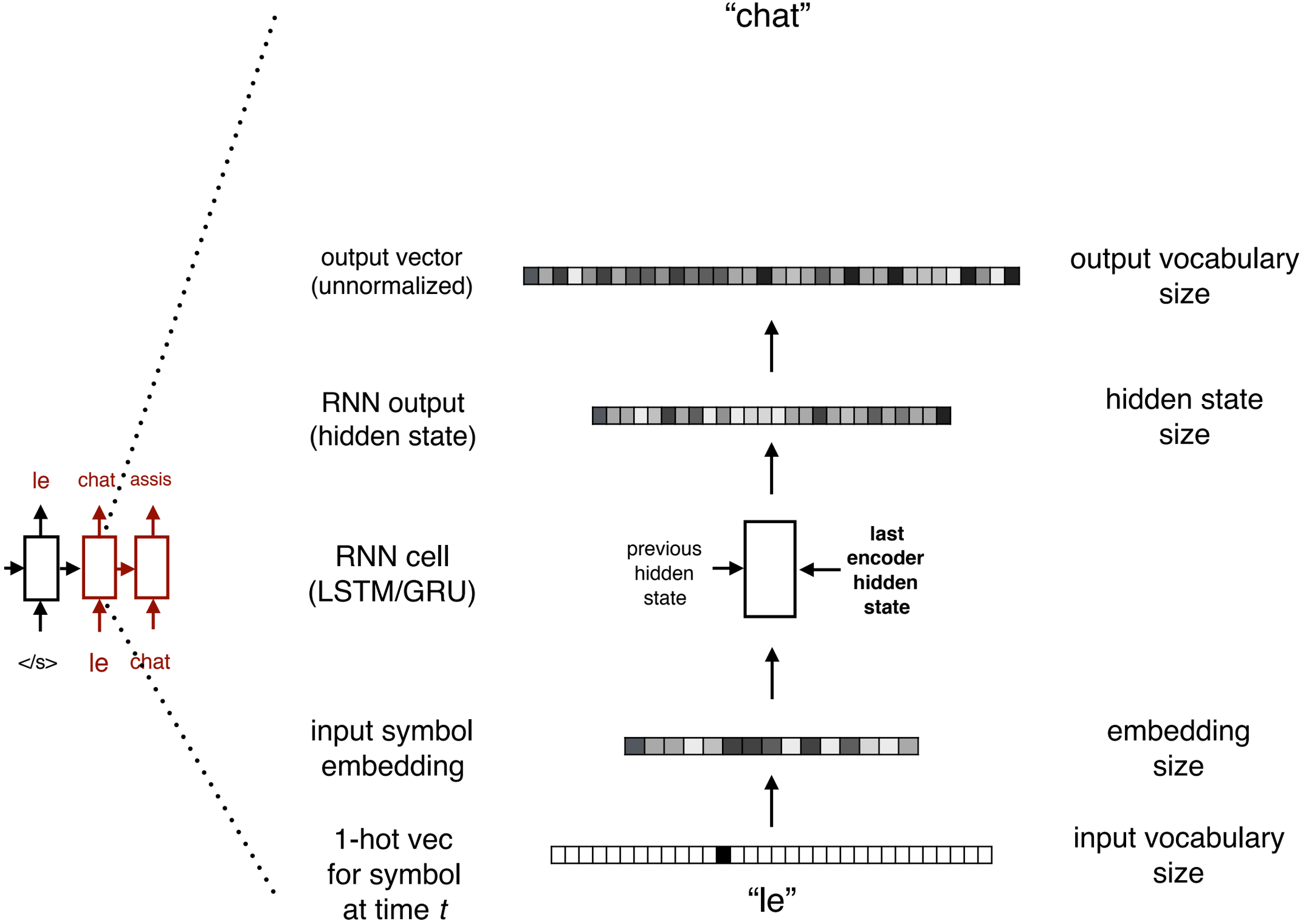
Seq2Seq decoder step - Zoom-In



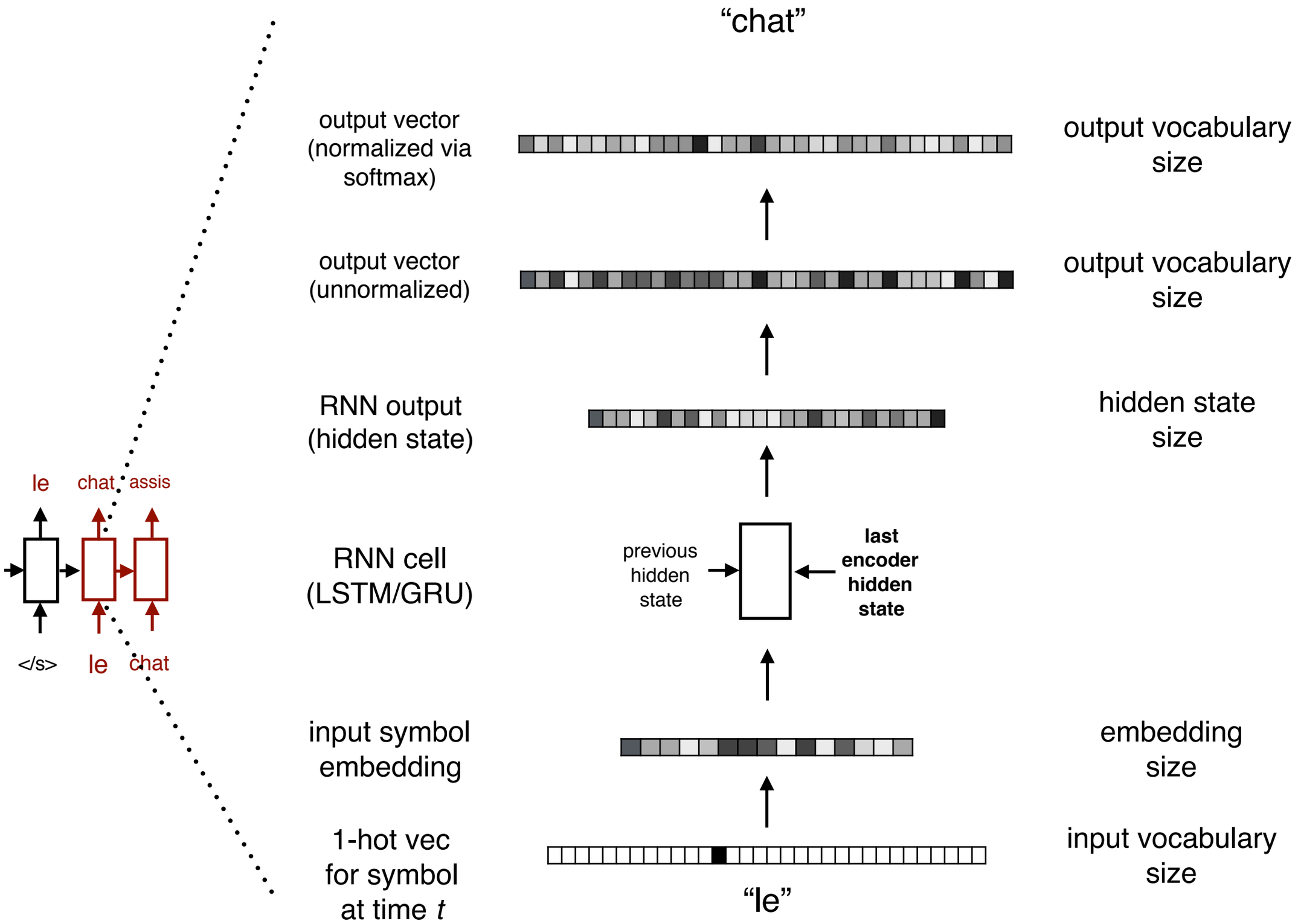
Seq2Seq decoder step - Zoom-In



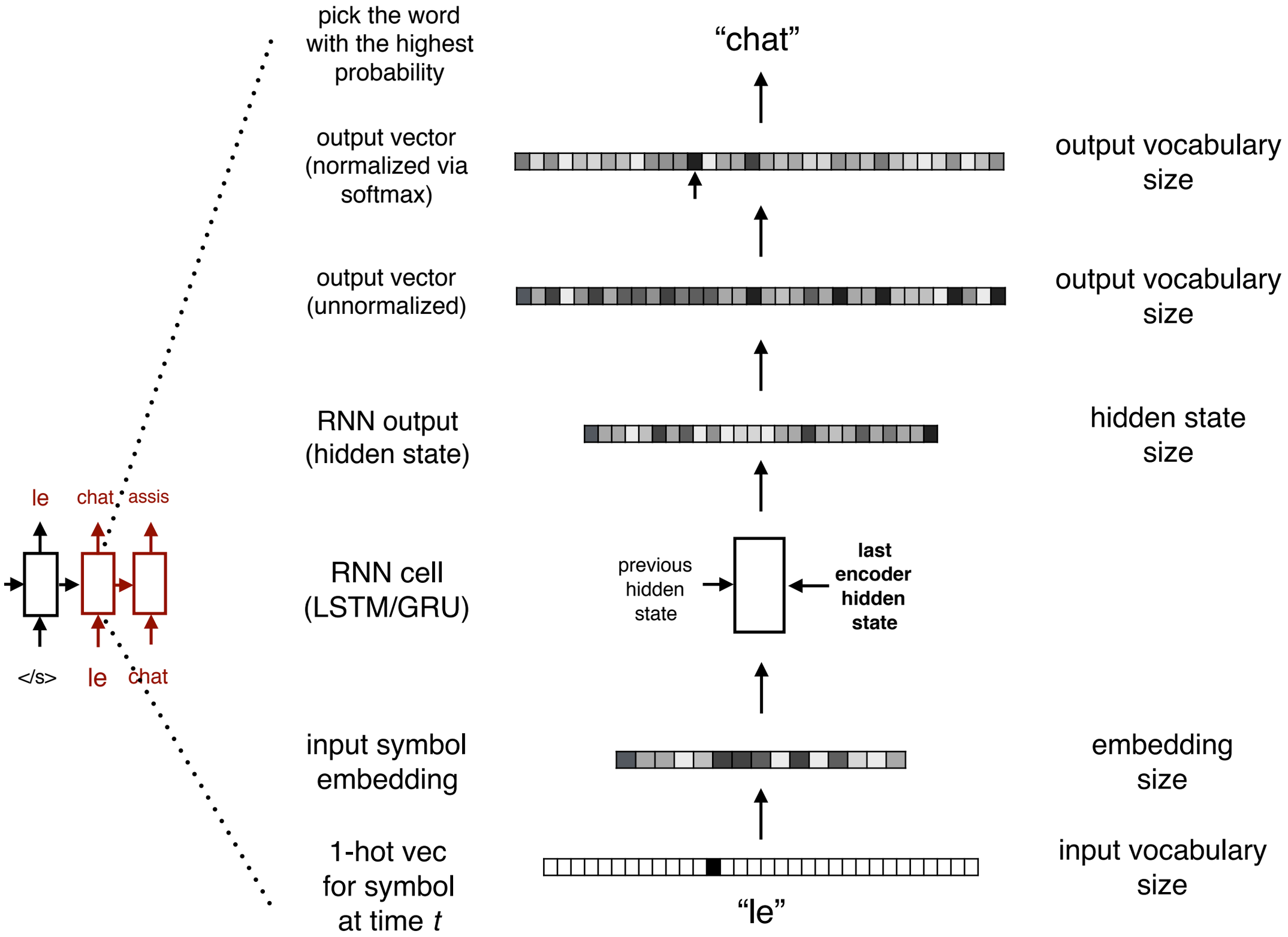
Seq2Seq decoder step - Zoom-In



Seq2Seq decoder step - Zoom-In



Seq2Seq decoder step - Zoom-In



The problem with vanilla seq2seq

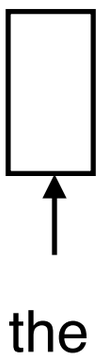
"You can't cram the meaning of a whole %&!$# sentence into a single $&!# vector!"* Ray Mooney

The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state

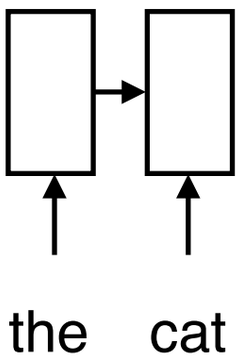
The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



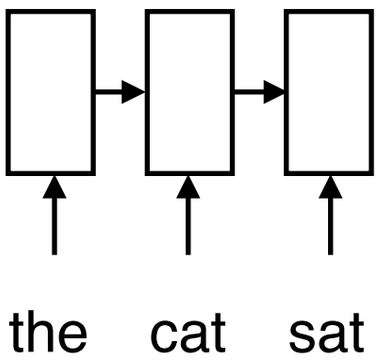
The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



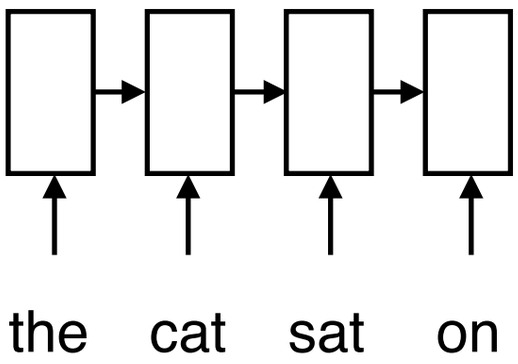
The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



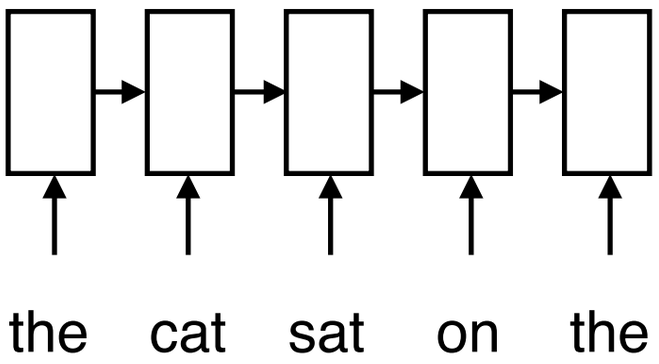
The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



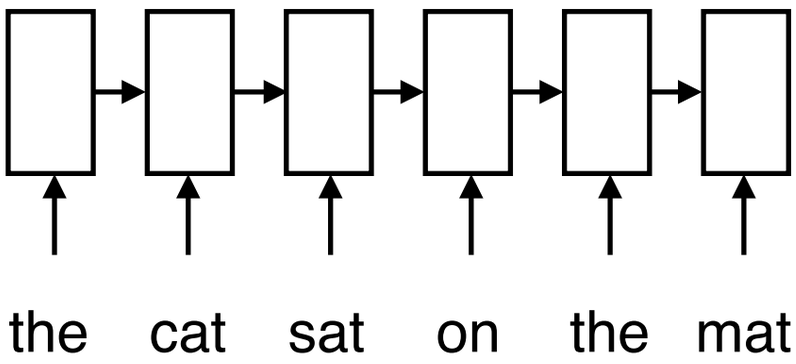
The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



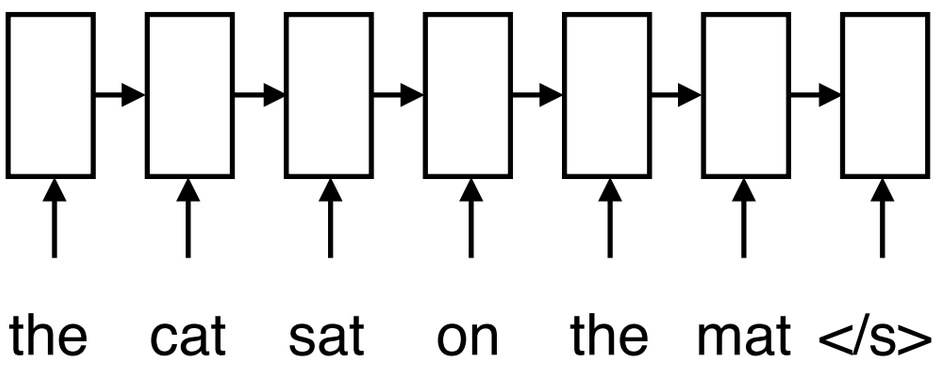
The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



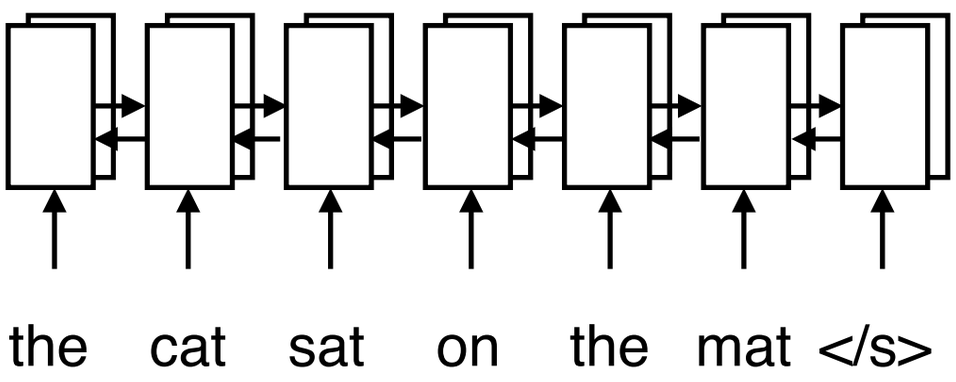
The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



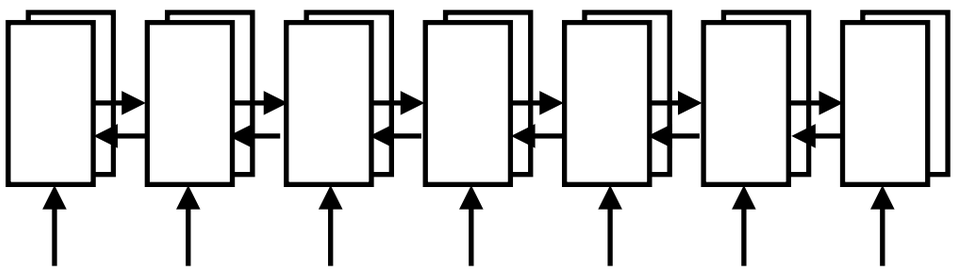
The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



The Attention Mechanism (this work)

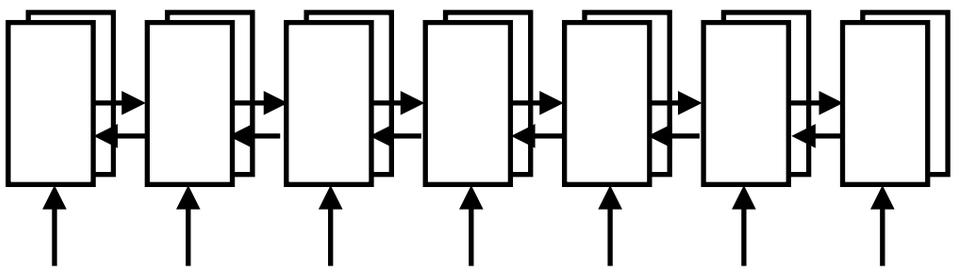
- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



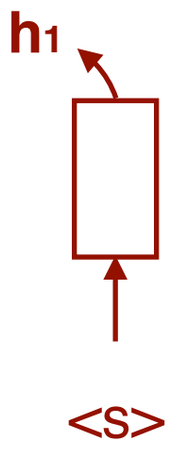
Bi-Directional Encoder

The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



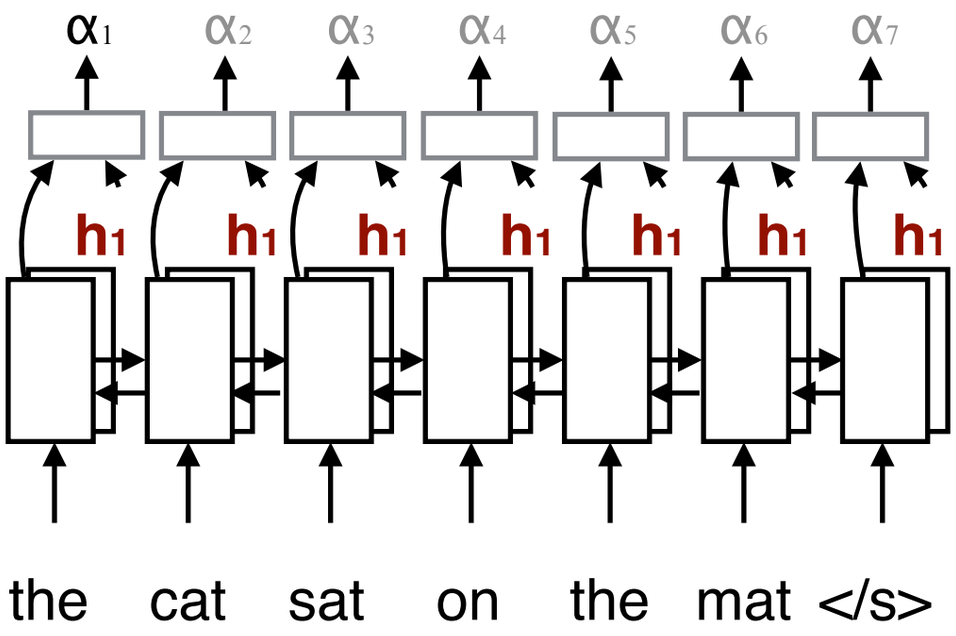
Bi-Directional Encoder



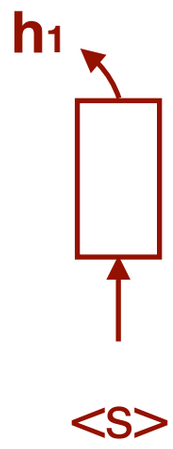
Attention-based Decoder

The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



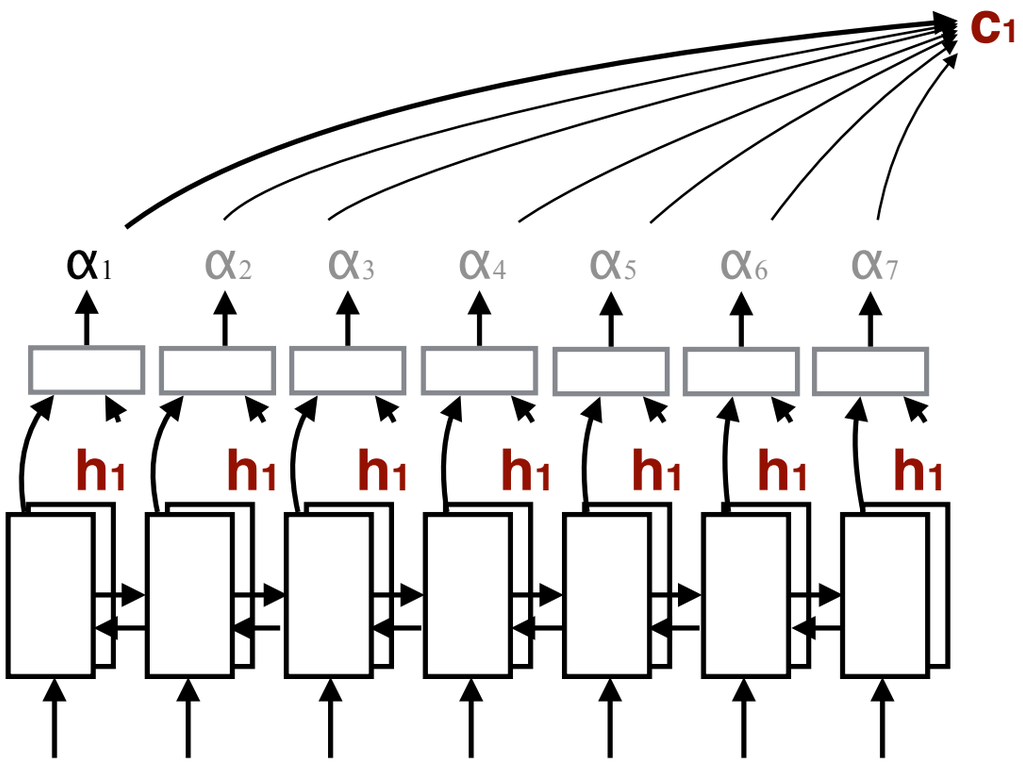
Bi-Directional Encoder



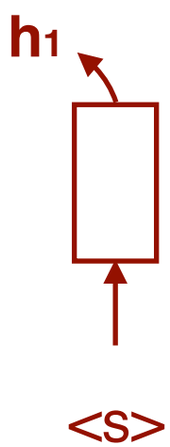
Attention-based Decoder

The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



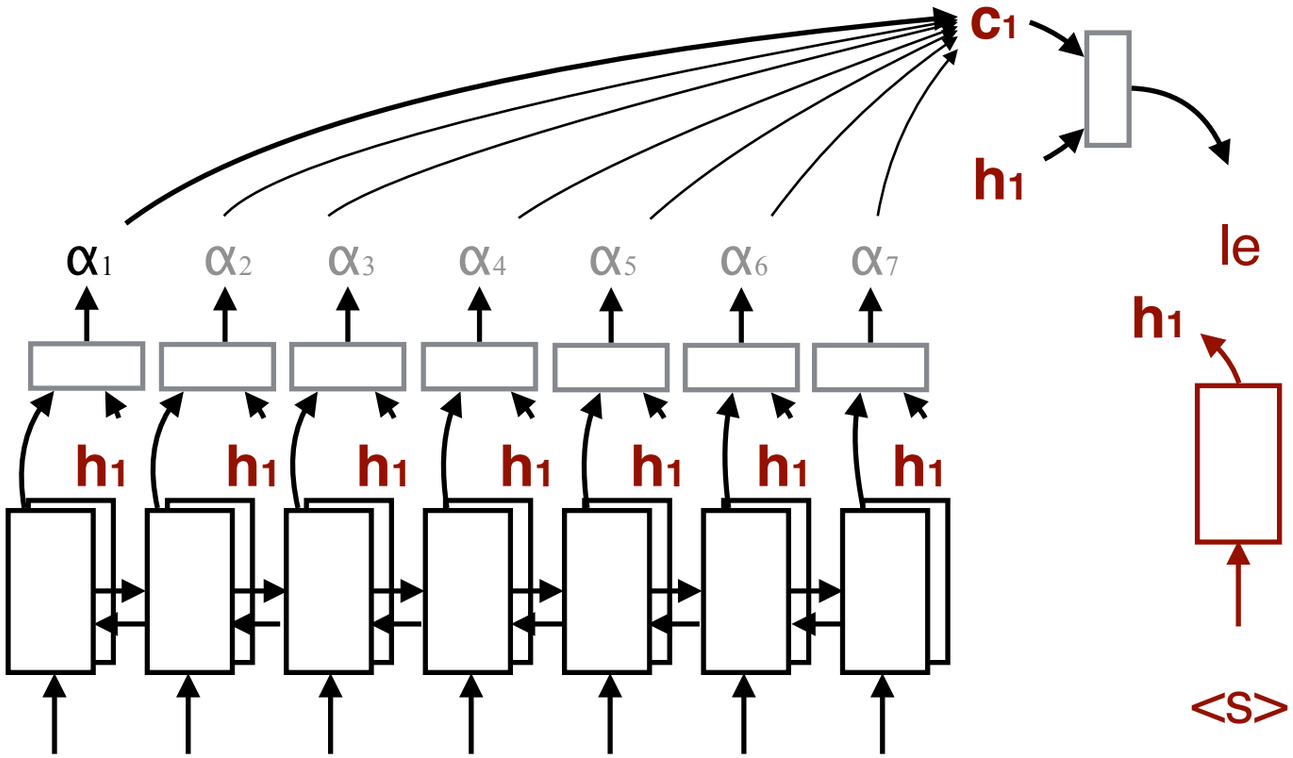
Bi-Directional Encoder



Attention-based Decoder

The Attention Mechanism (this work)

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state

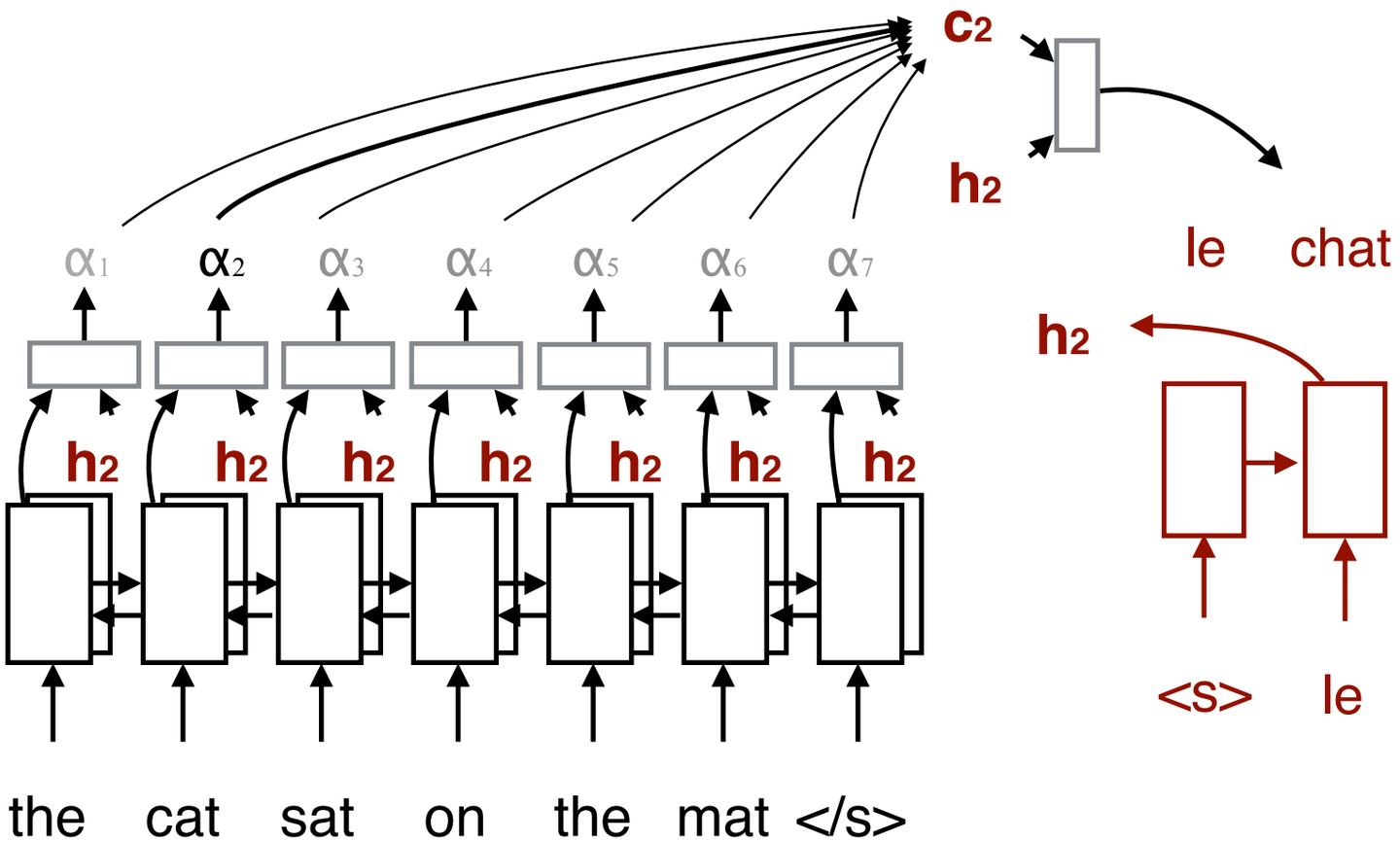


Bi-Directional Encoder

Attention-based Decoder

The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The importance of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state

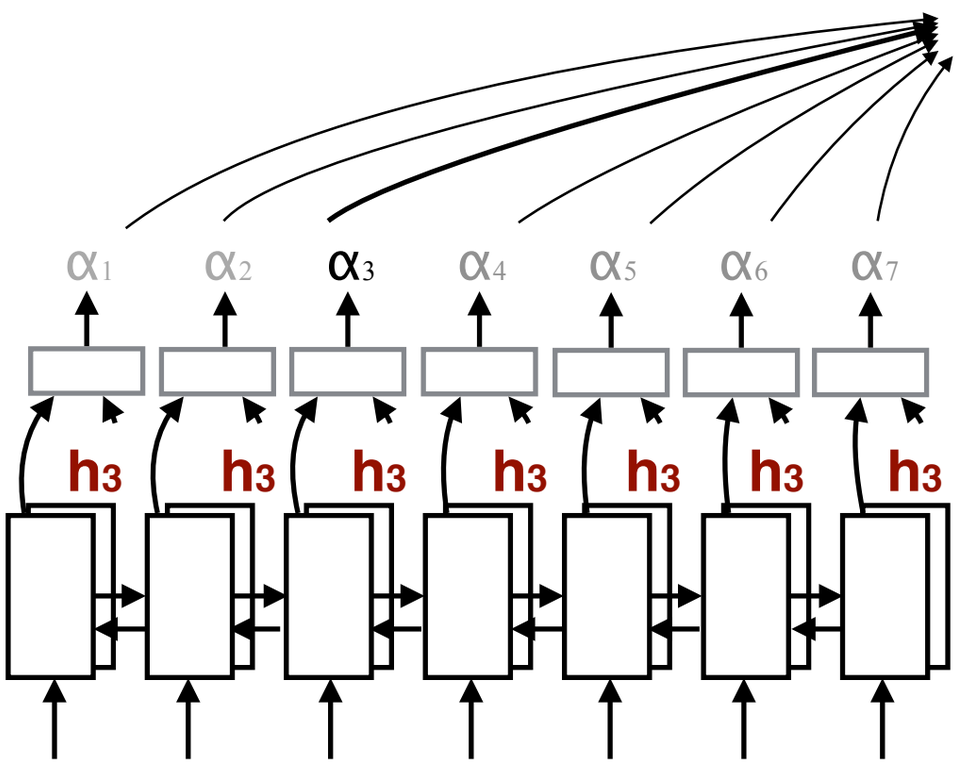


Bi-Directional Encoder

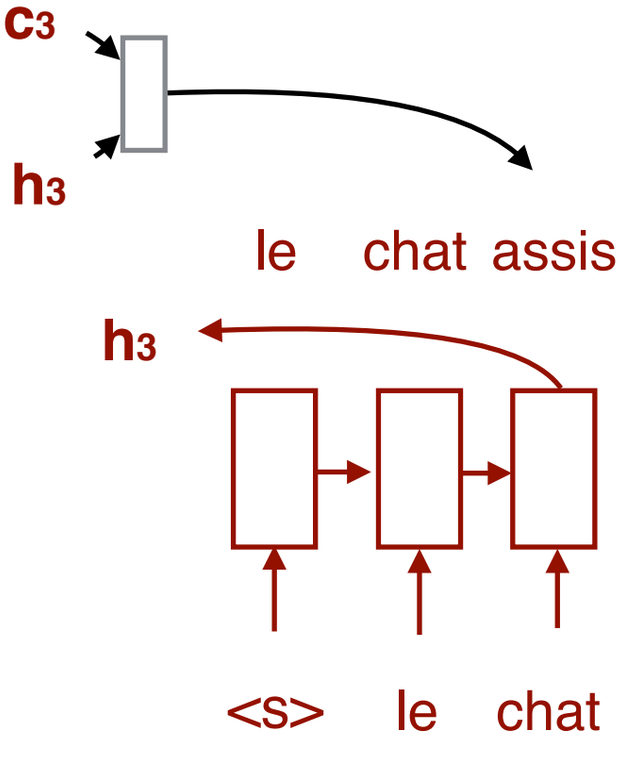
Attention-based Decoder

The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The importance of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



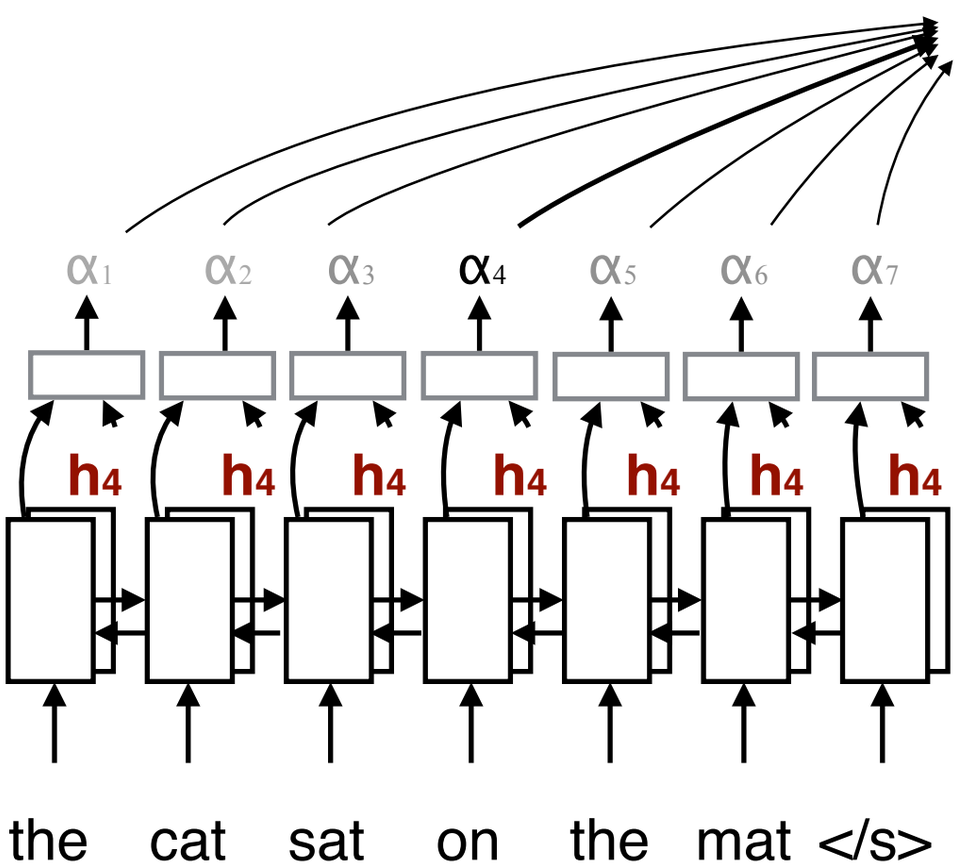
Bi-Directional Encoder



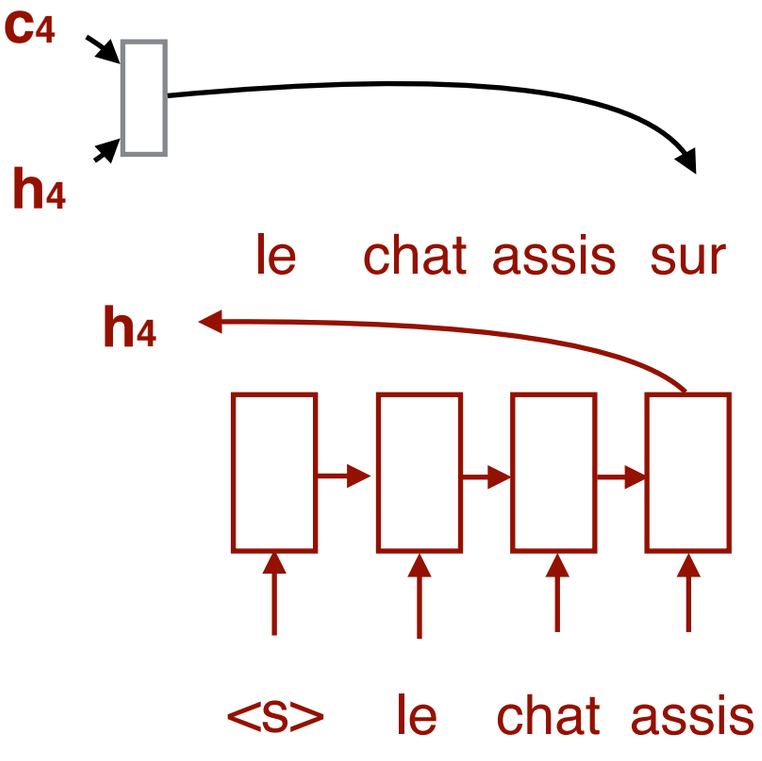
Attention-based Decoder

The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The importance of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



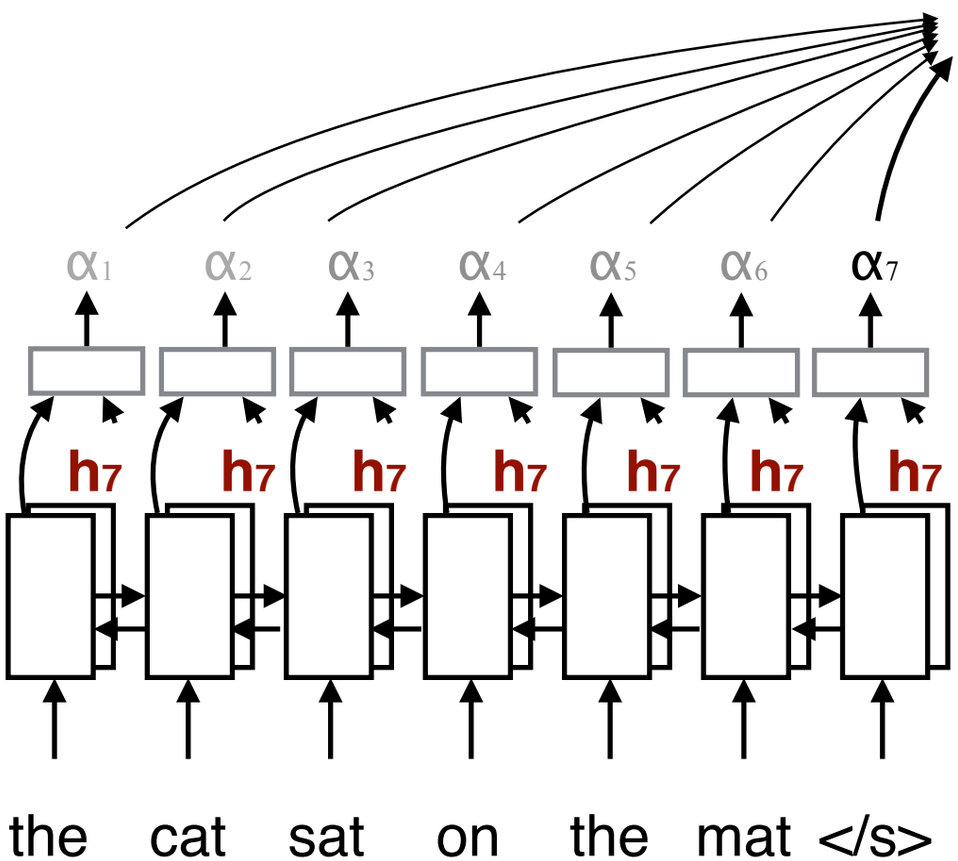
Bi-Directional Encoder



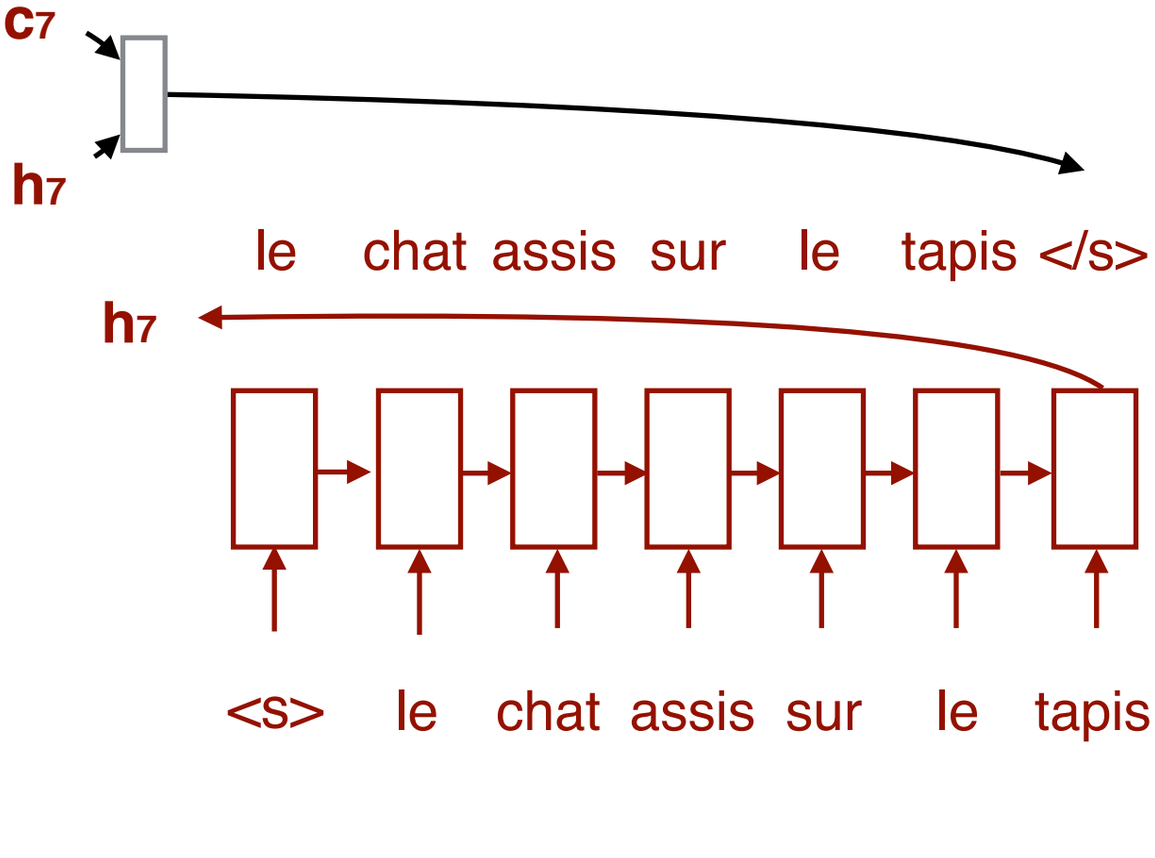
Attention-based Decoder

The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The importance of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



Bi-Directional Encoder



Attention-based Decoder

The Attention Mechanism

- And a bit more formally - in each decoder step:
  - Compute attention scores for each input element:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \tanh(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s])$$

  - Normalize the attention scores so they sum up to 1:

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

  - Compute c_t:

$$c_t = \sum_{j=1}^{T_x} a_j \bar{h}_j$$

  - Compute attention output state:

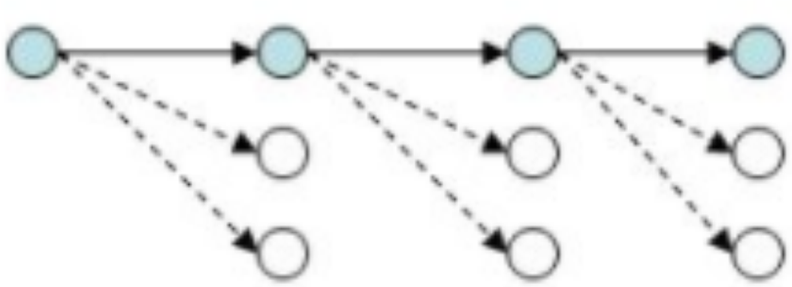
$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

  - Compute output probability distribution:

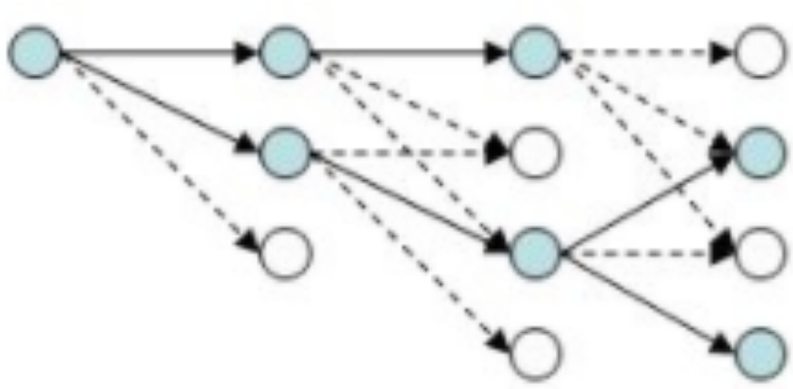
$$p(y_t | y_{<t}, x) = \text{softmax}(\boldsymbol{W_s}\tilde{\boldsymbol{h}}_t)$$

Decoding with Beam Search

- Instead of keeping one best option on each time step, keep k best options which are updated as-you-go
- Usually a small beam size is enough (5-12)



Greedy Search



Beam Search (k=2)

BLEU score

- The most common way to measure machine translation accuracy
- Based on n-gram precision

precision for each n-gram size (usually 1-4)

$$p_n = \frac{\sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}$$

Brevity Penalty - Small if candidate is too short

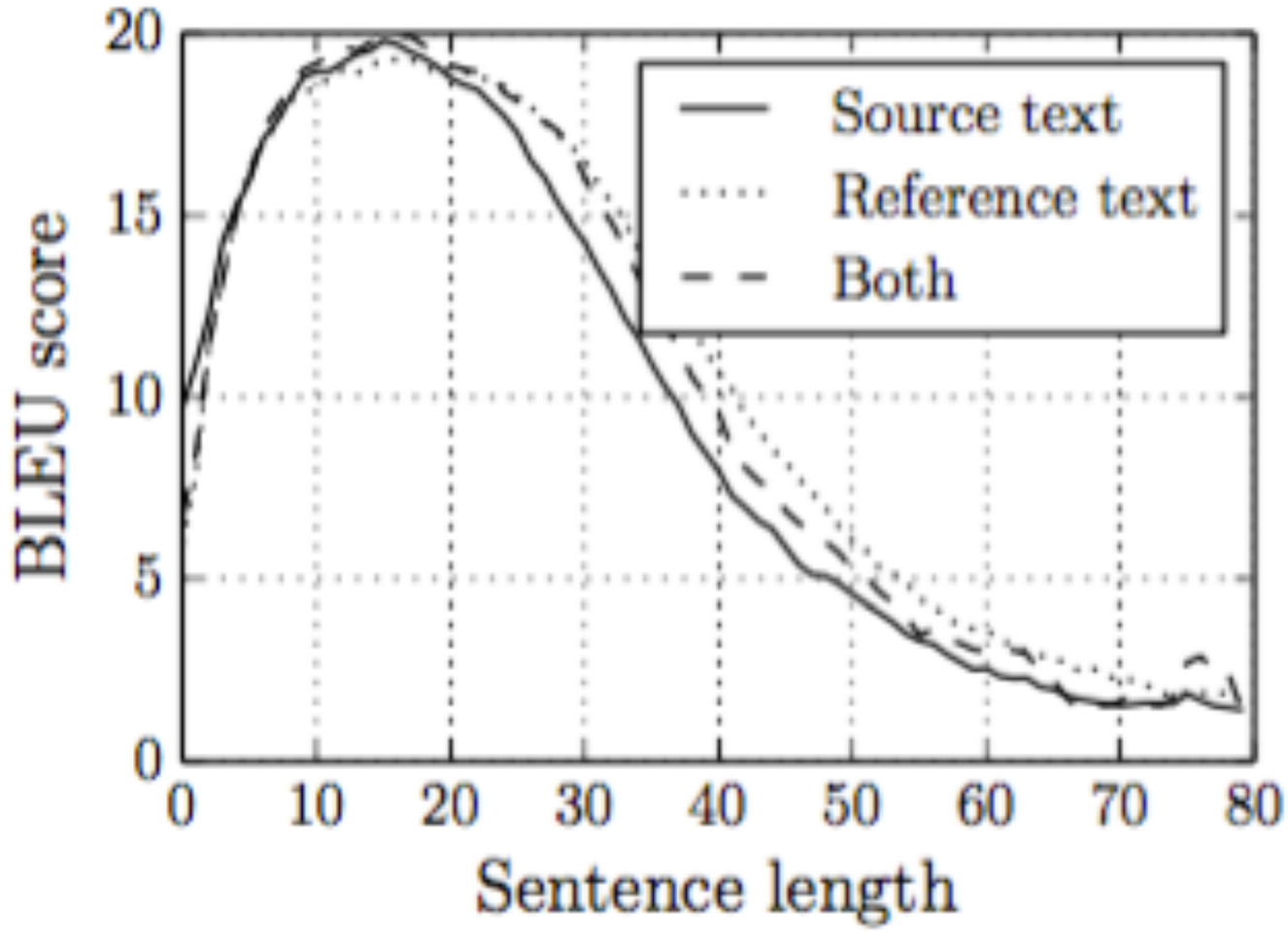
$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

BLEU score

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), w_n = 1/N$$

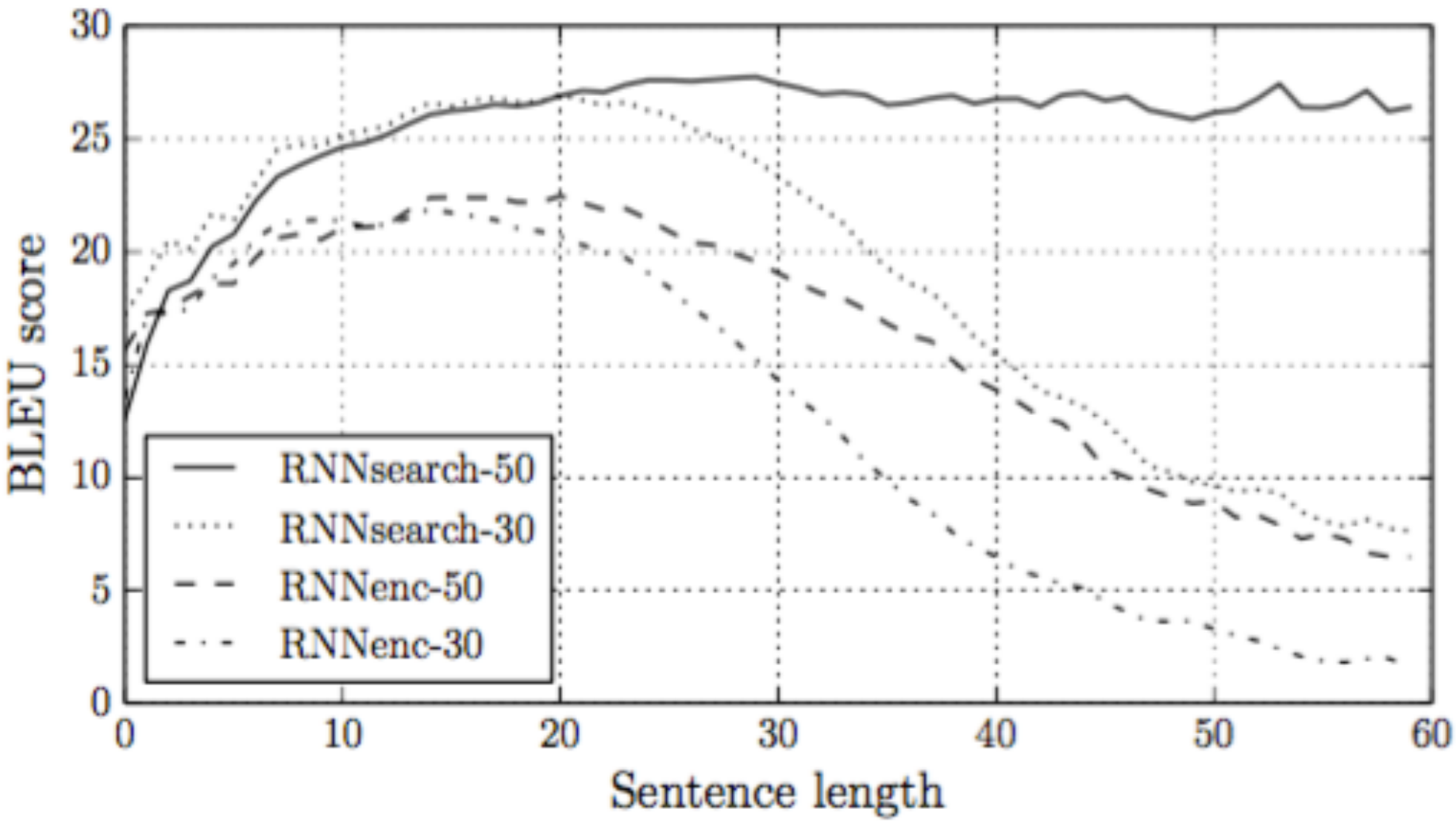
Results by Sentence Length - Before Attention

- Long sentences are very hard as they are "compressed" to a fixed length vector



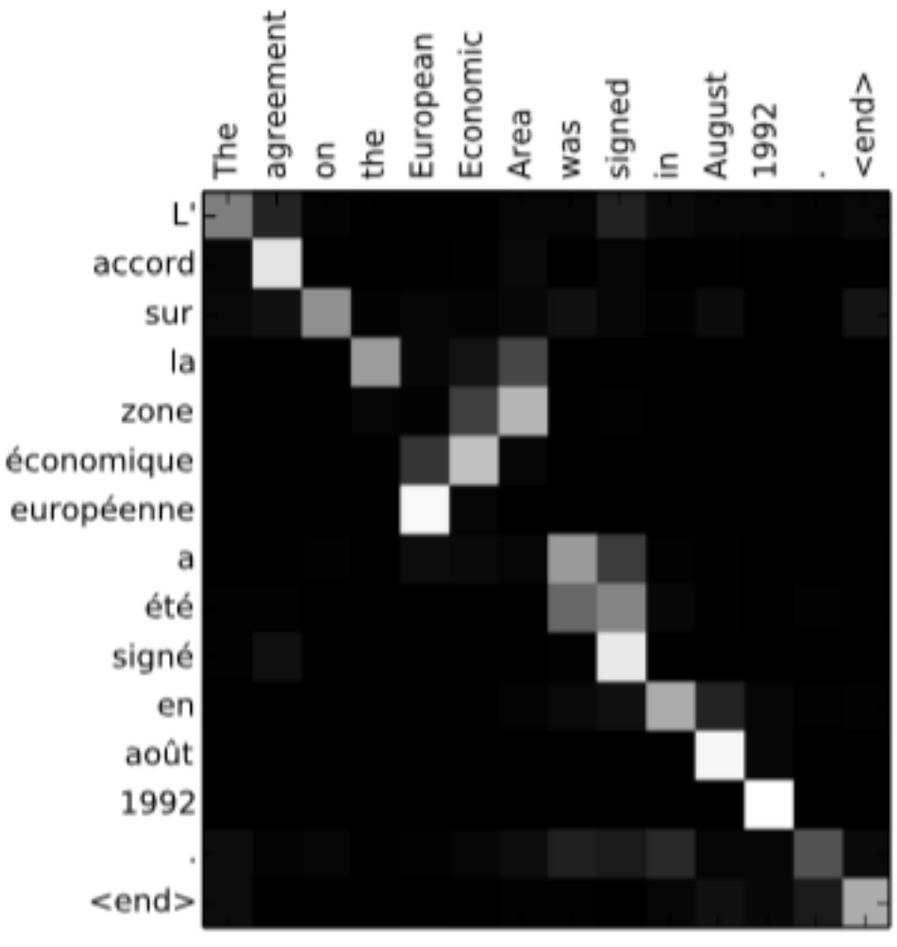
Results - After Attention

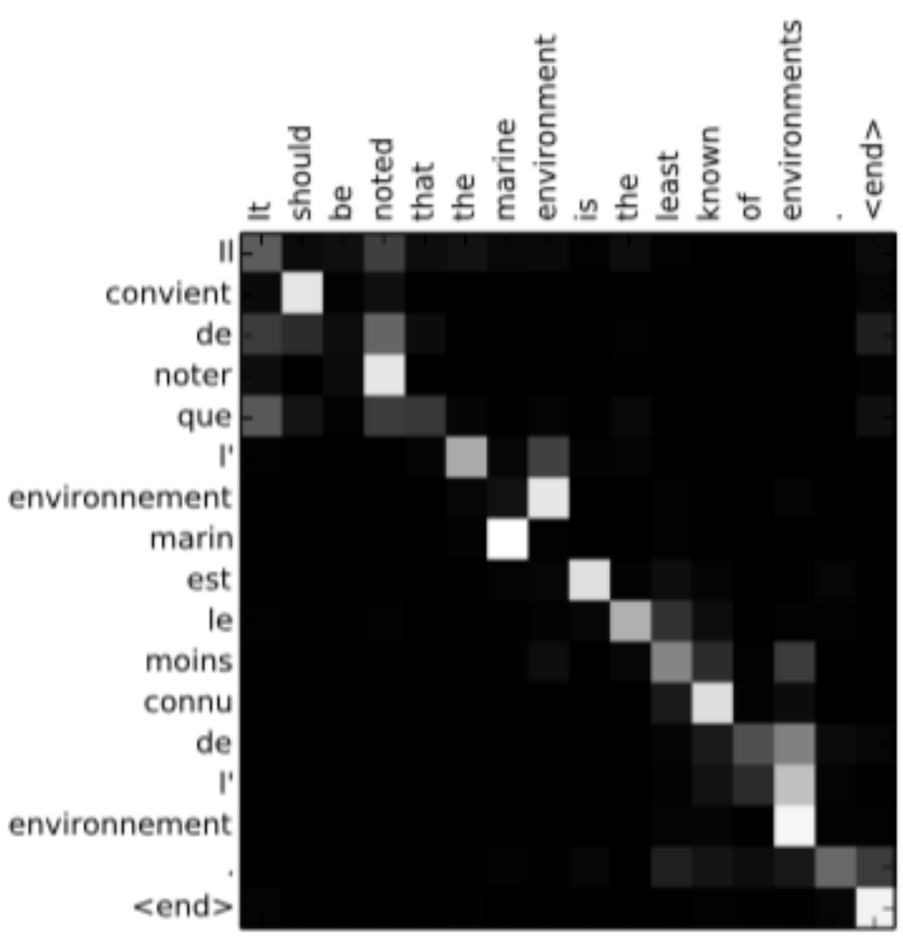
- The attention mechanism helps to overcome the issue



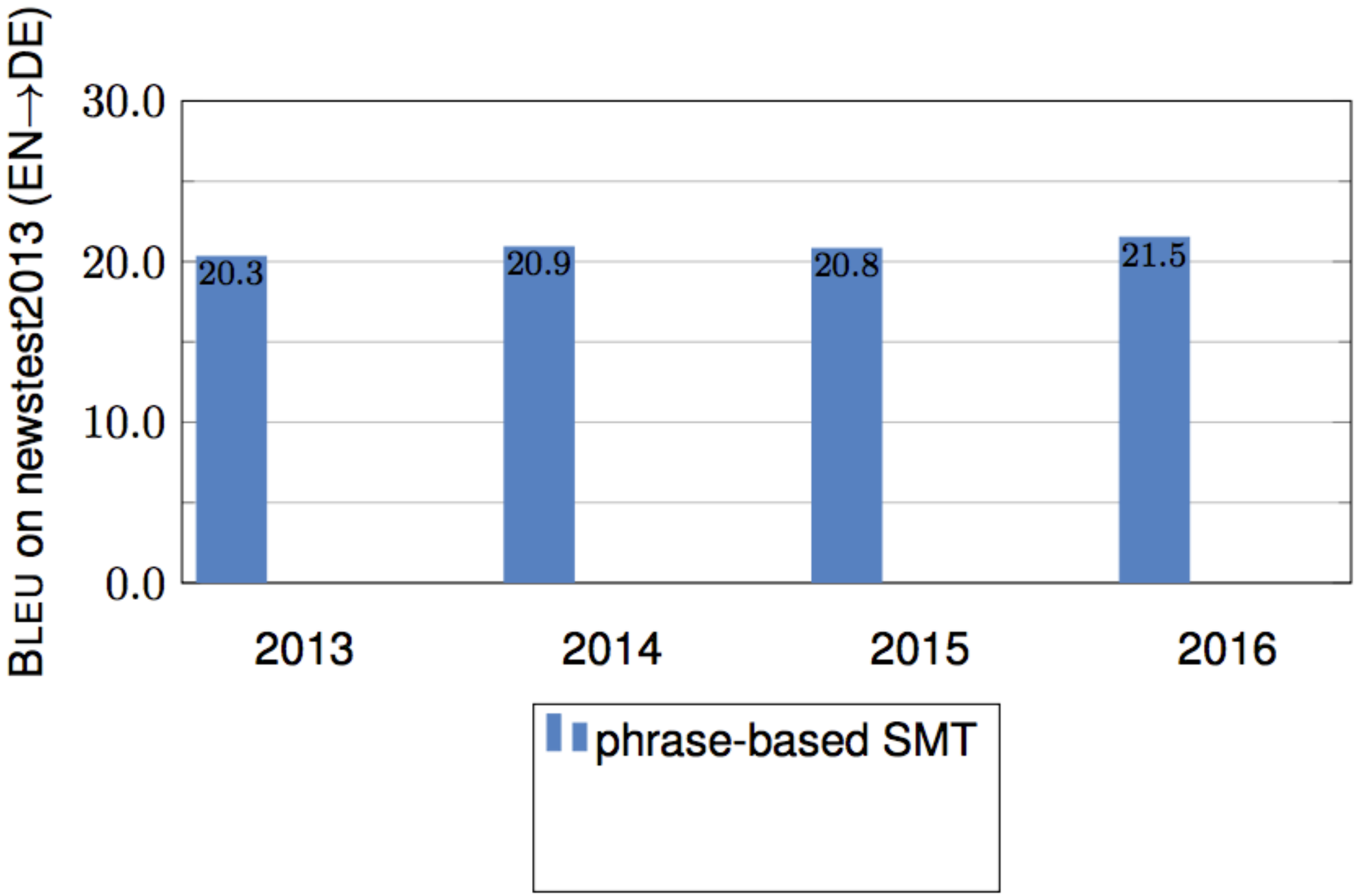
Results - After Attention

- The model learns nice alignments:

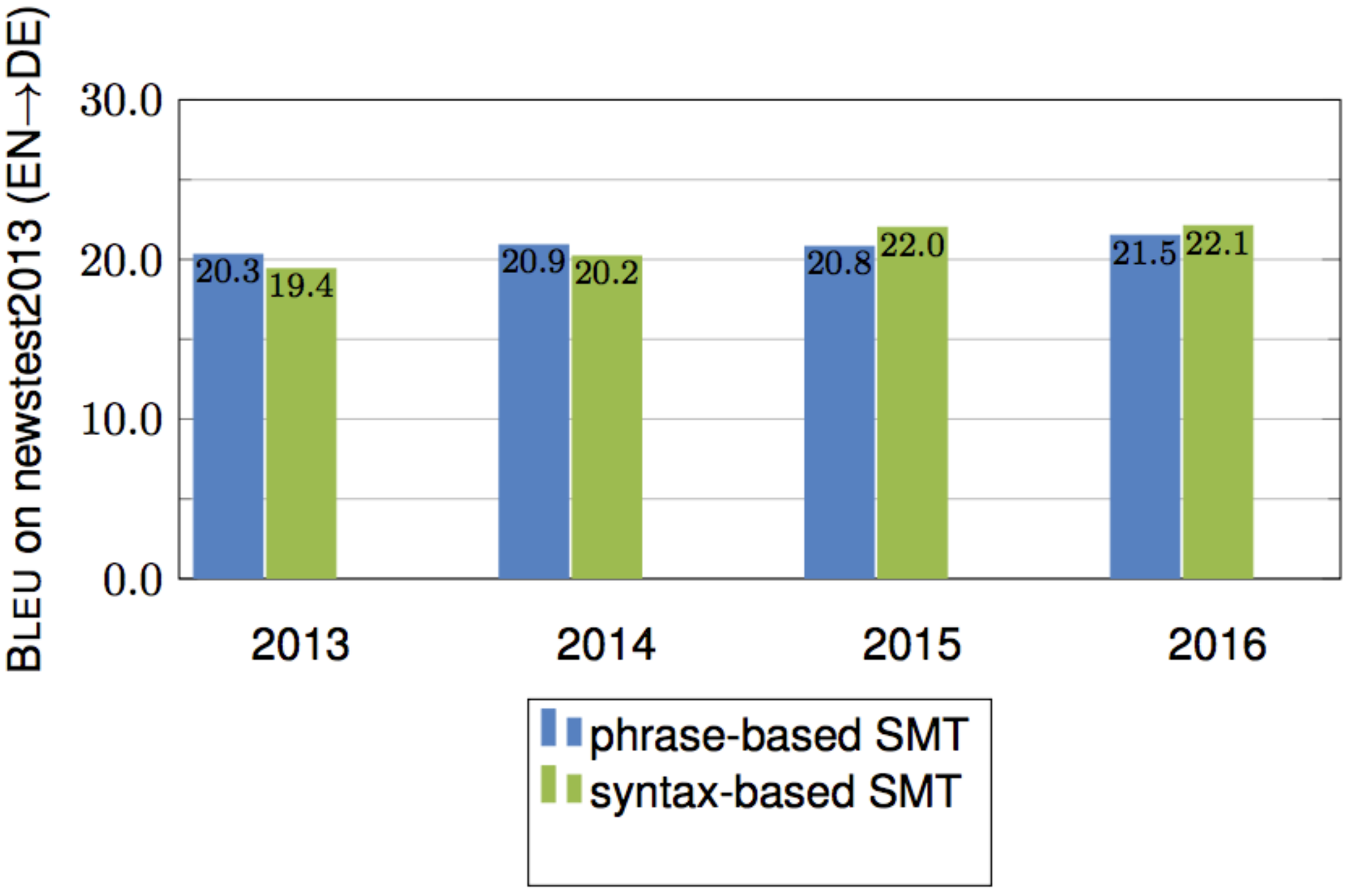




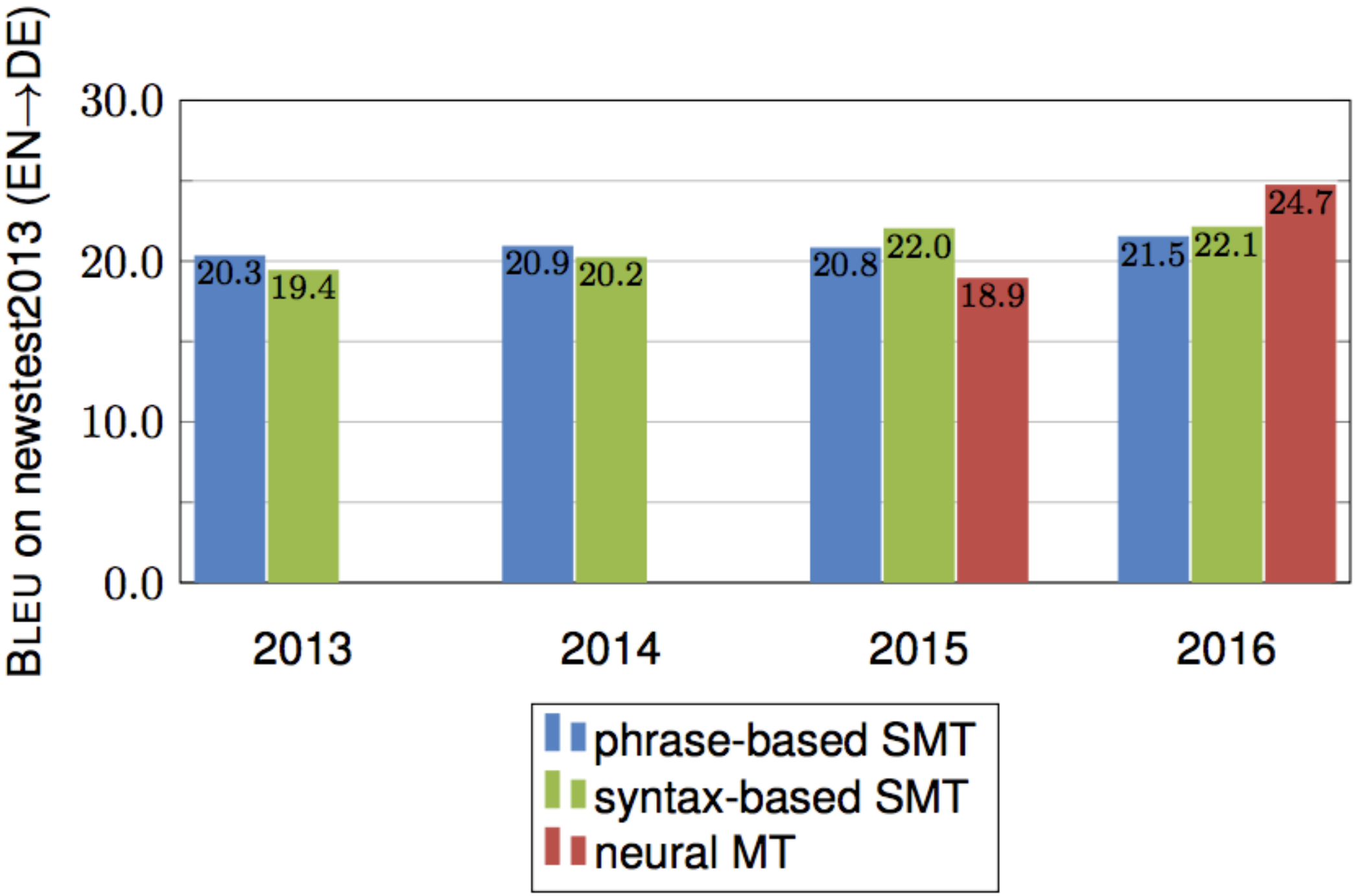
Edinburgh's* WMT results over the years



Edinburgh's* WMT results over the years

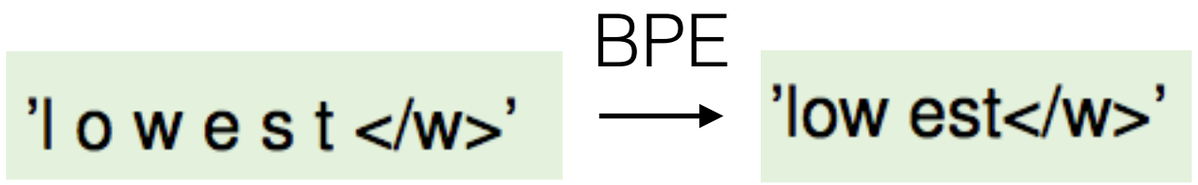


Edinburgh's* WMT results over the years

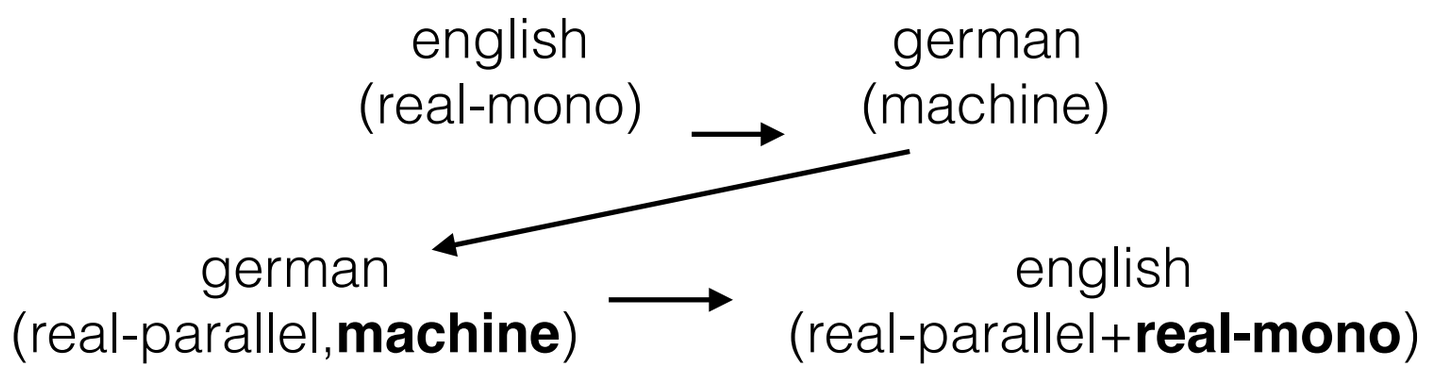


More Recent Improvements (Sennrich, 2016)

- BPE - work at sub-word level to create an open vocabulary



- Use monolingual data for training through back-translation



- Bi-directional Decoding:

a b c → x y z

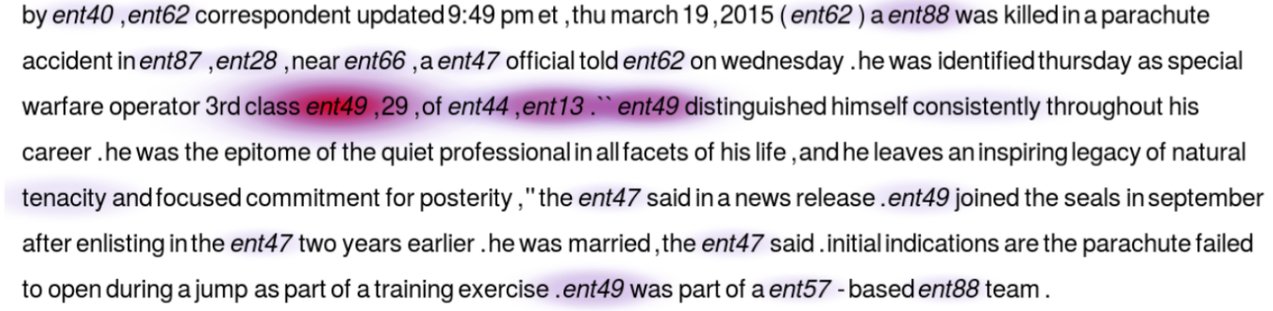
a b c → z y x

- And many others…

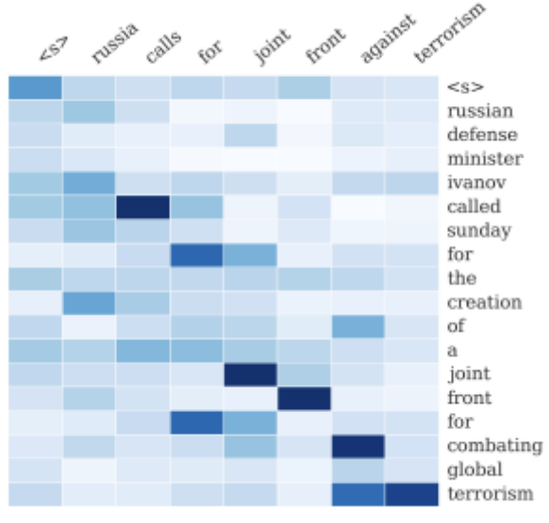
Attention in Other NLP tasks

- "BiLSTM's with attention seem to be taking over the field and improving our ability to do everything" (Chris Manning, 2016)

- Reading comprehension (Hermann et al., 2015; Blunsom, 2015)

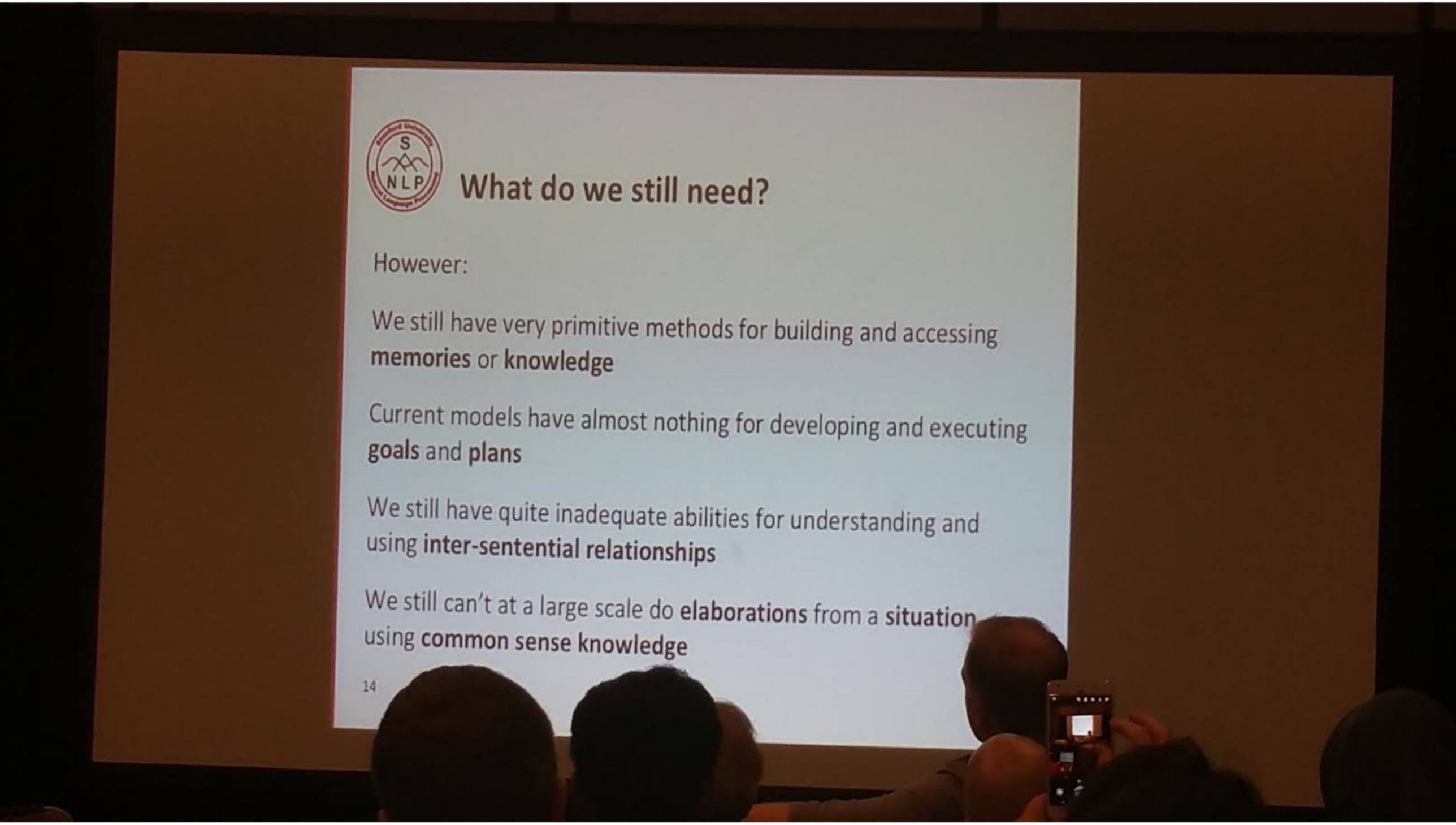


- Text Summarization (Rush et. al, 2015)



- And many more…

Summary


What do we still need?
However:
We still have very primitive methods for building and accessing memories or knowledge
Current models have almost nothing for developing and executing goals and plans
We still have quite inadequate abilities for understanding and using inter-sentential relationships
We still can't at a large scale do elaborations from a situation using common sense knowledge
14

Any Questions ?

Questions diverses ?

References

- A Primer on Neural Network Models for Natural - Yoav Goldberg Language Processing

- Neural - Yoshua Bengio ,Kyunghyun Cho ,Dzmitry Bahdanau Machine Translation by Jointly Learning to Align and Translate

- Thang) Stanford CS224N - Neural Machine Translation Talk (Luong

- K. Duh, Deep Learning Tutorial at DL4MT winter school

- Chris Olah, Understanding LSTM networks

- Rico Sennrich, NMT: Breaking the performance plateau